Commodore

# The Transactor

Commodore Canada's
Tech/News Periodical

VOLUME 3
Issue #1

## Bits and Pieces

### Mistaken Identity

The latest version of BASIC for Commodore computers is BASIC 4.0. The version before this was BASIC 2.0. However, this is often referred to as BASIC 3.0. There is no difference between the two. In actual fact, BASIC 3.0 was never released! A brief summary:

BASIC 1.0 : Original small keyboards PETs
Power-up message: *** COMMODORE BASIC ***
BASIC 2.0 : BASIC 1.0 upgrade and factory installed in big keyboard 2001 series machines
Power-up message: ### COMMODORE BASIC ###
BASIC 3.0 : Only difference from 2.0 was faster garbage collection. Never released.
BASIC 4.0 : Contains fast garbage collection and disk commands implemented in BASIC keywords. Factory installed in all 4000 and 8000 series PET/CBMs.
Power-up message: *** COMMODORE BASIC 4.0 ***

So BASIC 3.0 is BASIC 2.0 and vice versa... a simple case of mistaken identity.

One last note: The main logic board that came with original PETs (BASIC 1.0) has only enough ROM sockets to allow upgrade to BASIC 2.0 (which most, if not all, of you have done already). Upgrade to BASIC 4.0 on these boards would require one more socket; the $B000 socket. The only way to accomplish this would be to connect the ROM via the memory expansion port. Otherwise a new board is necessary. BASIC 4.0 upgrades are certainly possible on all other boards as the $B000, $A000 and $9000 sockets are provided.

### Software Portability

While on the subject of BASICs,... there have been several inquiries concerning the operation of Commodore software products from one BASIC version to the other. Visicalc and WordPro 3 will work on both BASIC 2.0 and BASIC 4.0. WordPro 3 is for 40 column machines, but it will actually work on the 80 column. It looks rather odd but it works! However, most won't have this combination.

Index Transactor #1, Volume3


Bits and Pieces ..... FP
  Mistaken Identity ..... FP
  Software Portability ..... FP
  NEC Again! ..... 2
  Microchess Conversion ..... 2
  8032 Reset Box ..... 3
  Montreal PET Club ..... 3
Backup ..... 4
PHS Contest Results ..... 5
The BMB String Thing ! ..... 7
DATA Line Generator ..... 12
BASIC Label Interface; Revisited ... 13
Review: The 8010 Modem ..... 15
File Spooling ..... 17
POKE OPEN Files ..... 19
Some Disk Utilities ..... 21
Cursor Coding ..... 24
Tax Ontario 1980 ..... 25
Bar Graph Printer ..... 29
Biocompatibilty ..... 33
DATA Positioning ..... 35
What's Available ..... 36
Print Using ..... 49

The Commodore Assembler Developement Pak is now delivered with both versions included. If you already have the BASIC 2.0 Pak, just send the original diskette (serial numbered label!) back to Commodore in Toronto and we'll return it to you with both versions.

As for other packages, the BASIC version number is included in the title of the software. If not, this implies correct operation on either BASIC 2 or 4.

NEC Again !

WordPro 4 Plus has a new command that uses the superscripting and subscripting features of the NEC Spinwriter. Set the internal switches of the NEC as follows. ( X = Do NOT Change)

| | | |
|---|---|---|
| back board : | SW1 = | 0 0 1 0 1 X 1 1 |
| 2nd from back board : | SW1 = | 0 0 0 0 0 0 0 0 |
| | SW2 = | 1 0 1 0 0 0 0 0 |
| | SW3 = | 1 0 1 0 0 0 0 0 |

Also see Pg 4, Transactor #12, and Pg 1, Transactor #11 (Volume 2) for more information.

MICROCHESS Conversion

As everyone knows, MICROCHESS 2.0 only runs on PET/CBMs that have BASIC 2.0 ROMs. For those that have upgraded to the new BASIC 4.0, the following sequence of commands will convert MICROCHESS for operation with the new ROMs.

```
1.  LOAD "MICROCHESS"                    ; use ',8' for disk,
                                           do not use DLOAD

2.  POKE 1055,0                          ; conversion done!

3.  SYS 4                                ; break to monitor

4.  .S "MICROCHESS 4",01,033A,2000       ;SAVE to Tape #1
                                           do not use Tape #2
OR
4.  .S "0:MICROCHESS 4",08,033A,2000     ;SAVE to disk drive
                                           drive #0
```

Some copies of Microchess don't allow re-SAVing this easily! If yours is one of them, just issue the above POKE each time before playing. Don't re-SAVE over your 2.0 version. Use a blank!

## 8032 Reset Box

Several programmers have aquired reset switches for their PETs to allow crash recovery and general resets without power-down. These switches usually come with a clip that connects to a line off the 555 timer on the rear right corner of the main logic board. On 40 column boards, this clip is placed on a resistor lead near the 555 timer as it tends to fall off the flat leads of the chip itself.

The arrangement of the 8032 boards is somewhat different. To use these reset switches on the 8032, locate the 555 timer at the rear right of the board. Just to the right and forward a bit is a diode marked "C50" on the board. The clip should be connected on the side of C50 that is closest to the 555.

## PET Club de la Montreal

Excuse my dreadful French. Last month I received a letter from K. H. King in Montreal. It read:

> Dear Karl J.,
>
> I would be most grateful if you would publish this letter or a condensed version in The Transactor.
>
> For some time now we have been without a PET Users Group in Montreal and I wonder if there are any PET owners who are still interested in having a club through which ideas can be exchanged, assistance given and in general support of all those who have a hobby interest in computers.
>
> If anyone is interested I would be very happy if they contacted me at the following address, or call me at (514) 842 7008 (home) or 844 6311 (Bus.)
>
> Keith H. King
> 3450 Drummond St. Apt 701
> Montreal, P.Q.
> H3G 1Y3

So c'mon you users with that wonderful accent!... Start a club! The Toronto Pet Users Group is now over 300 strong and still growing!

## Backup

## Screen Saver

In the last Transactor (Vol 2. #12, pg 3), a program was published that would store the contents of the screen in a PRG file on disk. Line 150 contains a bug a should be changed to:

```
150 PRINT#8, CHR$(PEEK(J));
```

Anyone using this technique probably caught this error.

## 8032 IRQ

On page 40 of the same issue, a method of disabling the window reset sequence was presented. It involved redirection of the IRQ interrupt vector so that the home count was always set to zero no matter how many times the HOME key were pressed during an INPUT statement. It seems that upon exiting the monitor, the IRQ vector is set back to normal, making step 4 a waste of time.

The code entered in step 3 works. Skip step 4, exit the montior, and change IRQ with:

```
POKE 144,122 : POKE 145,2
```

Set a window by positioning the cursor and hitting the 'Z', 'A' and 'L' keys simultaneously. Now try and clear it! (Sorry 'bout that Ken)

## ROM Entry Points

On page 34 of Transactor #12, an entry point table was published for all ROM versions. 16 lines from the bottom is a line that reads:

```
F7DC F7BC F7DF Set output device from LFN.
```

The BASIC 4.0 address (F7DF) should be changed to read F7FE. You might also want to change this on the same table on the reference page, centre of issue #12. (thanx Dave).

## Cross-Ref

In Transactor #9, Volume 2, Jim Butterfield published a Cross Reference program (page 18). Although nobody would think it possible, even Jim Butterfield can be hit by bugs (sorry Jim).

First erase lines 205 and 206. Now add " +L$ " to the end of line 200 and add line:

```
315 IF C2=6 AND LEN(M$)<5 THEN M$=" "+M$ : GOTO315
```

## PHS Contest Results

The winner of the PHS contest is:

Larry I. Miller
Regina, Saskatchewan

In our last issue we held a contest for the shortest routine to do a PHS; push the stack-pointer onto the stack. We would like to thank all who submitted entries but we've changed our minds about the winner. Instead we've decided that all those who entered will receive free subscriptions with an extra Volume 3 going to the address of Larry's choice. Larry's entry was:

```
 1.  08          PHP             ;dummy push
 2.  08          PHP             ;save flags
 3.  48          PHA             ;save .A (accumulator)
 4.  8A          TXA
 5.  48          PHA             ;save .X register
 6.  BA          TSX             ;stack pointer to .X
 7.  8A          TXA             ;and then to .A
 8.  18          CLC
 9.  69 04       ADC #04         ;restore SP value
10.  9D 04 01    STA $0104,X     ;store SP on stack
11.  68          PLA
12.  AA          TAX             ;restore .X
13.  68          PLA             ;restore .A
14.  28          PLP             ;restore flags
17 bytes, 41 cycles
```

> "All internal registers return with original contents. No memory is used except for the stack area. The value pushed onto the stack is equal to that of the Stack Pointer before the routine. At the end of the routine, SP is decremented like a true PUSH instruction."

A variation of the above was submitted by Chuan Chee of St. Catherines, Ontario. By reversing steps 5 and 6, the Stack Pointer goes in .X before the original contents of .X (now in .A) are pushed on the stack. This just means that .X contains a value one greater than in Larry's implementation. Remember, the Stack Pointer starts at $FF (See note 1). Every PUSH decrements SP. To compensate, steps 9 and 10 become:

```
 9.  69 03       ADC #03         ;restore SP value
10.  9D 03 01    STA $0103,X     ;store SP on stack
```

Other entries were shorter but incorrect. Many entries were correct but none as short and clean as 1st place. Nonetheless, free Volume 3's go out to:

Roger Burrows
Nepean, Ont.

David Berezowski
Thunder Bay, Ont.

Steve Punter
Mississauga, Ont.

James Yost c/o Eugenia Revas
Somerville, MA

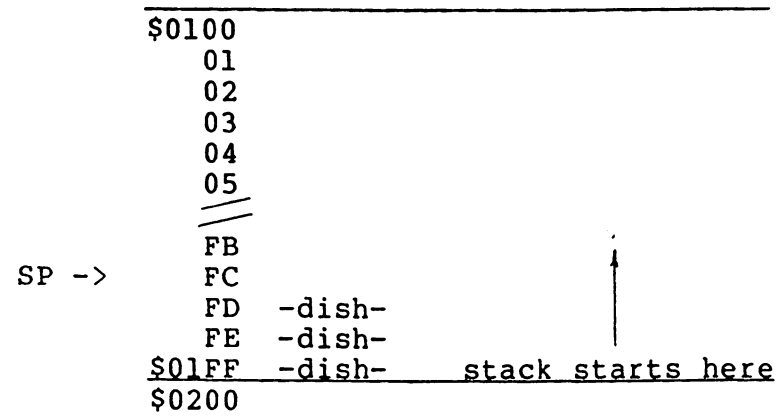
John Macdonald
Wawa, Ont.

???, behalf of L. Miller

All 1194 of the other subscribers entries must have been lost in the mail.

Note 1: The PET actually does a few PUSHes of its own during reset. SP decrements to $FA before the cursor flashes. Try reseting, then entering the monitor with SYS4. Notice SP at $F8 ? Now exit the monitor ('X' and RETURN) and re-enter it. SP is now two less than before. The SYS command puts a return address on the stack but exiting the M.L.M. does not take it off. Perhaps there is a reason for this but RUN or ?SYNTAX ERROR fixes everything.

If you picture stack memory space in memory map format, that is low addresses at the top, high addresses at the bottom, you'll see that it operates like "a stack". Think of it like a stack of dishes. SP points at the location where the next dish will go. Initially this is the very bottom of the stack ($FF). As dishes are piled on the stack, SP goes higher (towards the low addresses).




Of course the last dish on is the first dish off. This is called 'LIFO'; Last In First Out. But unlike dishes, the stack can only handle 256 entries so be careful. The stack pointer will "wrap around" if too many PUSHes are done. In this case the "dishes" at the bottom of the pile are replaced by new ones. If these were important return address dishes, you might find the whole machine comes crashing down.

Those that attempted the exercise probably see how virtually useless a PHS intruction is. Understanding the stack is the main objective and coding a PHS does that rather effectively!

## The BMB String Thing !

This is the utility you've been waiting for ! It's a combination of two previous BMB utilities plus a new one that's just fantastic! Due to some tricky string manipulations, this utility will only work on BASIC 4.0.

### 1. Block Get

This is the INPUT# statement substitute. INPUT up to 255 characters at one time. Works with any type file (SEQ, USR, REL). The best thing about using Block Get is that every character is retrieved; leading spaces, quotes, commas, colons and even CHR$(0). Two back-to-back carriage returns will crash the INPUT# command but this doesn't bother Block Get!

Input stops on a carriage return (CHR$(13)). This can be changed to any other character that you wish to use as a delimeter i.e. binary 13 might represent some other data. EOI will also terminate input which is handy for REL file use i.e. trailing carriage return not required. If EOI comes in with a CHR$(0) the last character is chopped off.

In SEQuential file operations you don't even need carriage returns (not suggested!). Block Get will continue getting characters until it has 255. At this point input stops allowing you to juggle the string. The next Block Get call starts with the next character from the file and goes again to 255 or EOI, whichever comes first. Remember to check ST after each call as is normal when INPUTting.

A 257 byte buffer is required for BASIC 4.0 string structure.

Format:

```
10  A$ = ""
20  DOPEN#8, "SOME FILE"
30  SYS 32514,8,A$
40  PRINT A$
50  IF ST = 0 THEN 30
60  DCLOSE #8
```

A$ need only be initialized once, eg. beginning of program.

### 2. Instring

Insert one string within another at position specified. The programmer is responsible for insuring that the string being inserted is not longer than the place it's going to.

To insert A$ into B$ at position 10:

```
SYS 32517 A$, B$, 10
```

### 3. Position Search

This one is fabulous. Search for the occurrence of one string within another string. If found, the position is placed in location 0; if not found, PEEK(0) = 0. Pattern matching capabilities (like the Commodore disks) can easily be incorporated into this routine.

Format is: search for any occurence of A$ in B$

```
1.   A$ = "ABCDE"
2.   B$ = "ABCDEFGHIJKLM"
3.   SYS 32520, A$, B$
4.   PRINT PEEK(0)
5.   A$ = "FGHIJ"
6.   SYS 32520, A$, B$
7.   PRINT PEEK(0)
```

Try this using direct commands. Note: No other string operators are allowed after the SYS (i.e. MID$, LEFT$, STR$, etc.).

## Preparing For The BMB String Thing

You should already have set up the utility before anything else is done. Since the utility sits in high memory, it must be sealed off so that BASIC can't clobber it.

```
POKE 53,126 : CLR
```

This POKE brings the top-of-memory pointer down, protecting it from other string operations. This BMB String Thing works in address space from $7E00 to $7FFF. If you have an assembler, type in the mere 3 pages of source and you can move it anywhere.

```
1000 FORJ= 32514 TO 32767 :READ X:POKE J,X:NEXT
1001 POKE53,126 : CLR :REM ROUTINE PROTECTED
1008 DATA  76, 107, 127,  76, 180, 127,  76, 211
1016 DATA 127,  32, 245, 190,  32, 152, 189, 160
1024 DATA   0,  96,  32,  11, 127, 177,  68, 133
1032 DATA   0, 200, 177,  68, 133,   1, 200, 177
1040 DATA  68, 133,   2,  32,  11, 127, 177,  68
1048 DATA 133, 136, 200, 177,  68, 133, 137, 200
1056 DATA 177,  68, 133, 138,  96,  56, 165, 138
1064 DATA 197,  47, 144,  28, 165, 137, 197,  46
1072 DATA 144,  22, 165, 136,  24, 101, 137, 133
1080 DATA 137, 144,   2, 230, 138, 160,   1, 165
1088 DATA 136, 145, 137, 200, 169, 255, 145, 137
1096 DATA  96,  32,  11, 127,  32,  45, 201, 165
1104 DATA  18, 240,   3,  76,   0, 191, 165,  17
1112 DATA  96,  32,  91, 127, 133, 210, 169,   0
1120 DATA 133,   1, 169, 126, 133,   2,  32,  37
1128 DATA 127,  32,  55, 127, 166, 210,  32, 198
1136 DATA 255,  32, 228, 255, 170, 201,  13, 240
1144 DATA  21, 160,   0, 145,   1, 230,   1, 164
1152 DATA   1, 192, 255, 240,   4, 164, 150, 240
1160 DATA 232, 138, 208,   2, 198,   1,  32, 204
1168 DATA 255, 160,   0, 165,   1, 145,  68, 200
1176 DATA 169,   0, 145,  68, 200, 169, 126, 145
1184 DATA  68,  96,  32,  20, 127,  32,  91, 127
1192 DATA 198,  17, 165, 137,  24, 101,  17, 133
1200 DATA 137, 144,   2, 230, 138, 160,   0, 177
1208 DATA   1, 145, 137, 200, 196,   0, 208, 247
1216 DATA  96,  32,  20, 127, 169,   0, 170, 133
1224 DATA 182, 160,   0, 177,   1, 209, 137, 208
1232 DATA  12, 200, 196,   0, 208, 245, 230, 182
1240 DATA 165, 182, 133,   0,  96, 230, 137, 208
1248 DATA   2, 230, 138, 230, 182, 165, 182, 197
1254 DATA 136, 208, 222, 134,   0,  96
```

```
BMBSTRINGTHING......PAGE 0001

LINE# LOC    CODE          LINE


0001  0000                 ;GENERAL PURPOSE ROUTINES FOR THE PET/CBM SYSTEM
0002  0000                 ;REQUIRES A 257 CHARACTER BUFFER
0003  0000                 ;STARTING AT A PAGE BOUNDARY
0004  0000                 ;
0005  0000                 ;FOR 32K BASIC 4.0 ONLY
0006  0000                 ;
0007  0000                 CHKCOM = $BEF5          ;CHECK FOR COMMA
0008  0000                 VARPAR = $BD98          ;EVALUATE EXPRESSION
0009  0000                 FLTINT = $C92D          ;FLOATING PT TO INT
0010  0000                 SYNERR = $BF00          ;?SYNTAX ERROR
0011  0000                 LEN1   = $00            ;WORK SPACE
0012  0000                 LEN2   = $88            ;     ''
0013  0000                 TEMP1  = $01            ;     ''
0014  0000                 TEMP2  = $89            ;     ''
0015  0000                 VARPTR = $44            ;CURRENT VARAIBLE POINTER
0016  0000                 INTEG  = $11            ;INT VALUE FOR SYS
0017  0000                 CURFIL = $D2            ;CURRENT FILE NUMBER
0018  0000                 BUFFER = $7E00          ;BLOCK GET BUFFER, MUST BE SEALED
0019  0000                 ;                        ;OFF WITH POKE53,126:CLR
0020  0000                 STATUS = $96            ;ST STORAGE
0021  0000                 ARYEND = $2E            ;END OF ARRAYS
0022  0000                        * = $7F02        ;STARTS 2 BYTS IN DUE TO BUFFER
0023  7F02                 ;
0024  7F02                 ; VECTOR TABLE
0025  7F02  4C 6B 7F              JMP BLKGET       ;BLOCK GET  - SYS 32514
0026  7F05  4C B4 7F              JMP INSTRG       ;INSTRING   - SYS 32517
0027  7F08  4C D3 7F              JMP POSTRG       ;POS SEARCH - SYS 32520
0028  7F0B                 ;
0029  7F0B  20 F5 BE       CHKPAR JSR CHKCOM       ;CHECK FOR COMMA AND
0030  7F0E  20 98 BD              JSR VARPAR       ;SET UP POINTER TO STRING
0031  7F11  A0 00                 LDY #00
0032  7F13  60                    RTS
0033  7F14  20 0B 7F       FINDAB JSR CHKPAR       ;FIND FIRST STRING
0034  7F17  B1 44                 LDA (VARPTR),Y
0035  7F19  85 00                 STA LEN1         ;STORE STRING 1 LENGTH
0036  7F1B  C8                    INY
0037  7F1C  B1 44                 LDA (VARPTR),Y
0038  7F1E  85 01                 STA TEMP1        ;STRING 1 ADDRESS LOW
0039  7F20  C8                    INY
0040  7F21  B1 44                 LDA (VARPTR),Y
0041  7F23  85 02                 STA TEMP1+1      ;STRING 1 ADDRESS HI
0042  7F25  20 0B 7F       FINDB  JSR CHKPAR       ;FIND ANOTHER STRING
0043  7F28  B1 44                 LDA (VARPTR),Y
0044  7F2A  85 88                 STA LEN2         ;STORE STRING 2 LENGTH
0045  7F2C  C8                    INY
0046  7F2D  B1 44                 LDA (VARPTR),Y
0047  7F2F  85 89                 STA TEMP2        ;STRING 2 ADDRESS LOW
0048  7F31  C8                    INY
0049  7F32  B1 44                 LDA (VARPTR),Y
0050  7F34  85 8A                 STA TEMP2+1      ;STRING 2 ADDRESS HI
0051  7F36  60                    RTS
0052  7F37  38             KILSTR SEC              ;GET RID OF OLD STRING
0053  7F38  A5 8A                 LDA TEMP2+1
0054  7F3A  C5 2F                 CMP ARYEND+1     ;STRING IN TEXT?
0055  7F3C  90 1C                 BCC NOKILL       ;YES, EXIT
```

```
LINE#  LOC   CODE        LINE

0056  7F3E  A5 89           LDA TEMP2
0057  7F40  C5 2E           CMP ARYEND
0058  7F42  90 16           BCC NOKILL
0059  7F44  A5 88           LDA LEN2
0060  7F46  18              CLC
0061  7F47  65 89           ADC TEMP2
0062  7F49  85 89           STA TEMP2
0063  7F4B  90 02           BCC NOHIIN
0064  7F4D  E6 8A           INC TEMP2+1
0065  7F4F  A0 01     NOHIIN LDY #01
0066  7F51  A5 88           LDA LEN2
0067  7F53  91 89           STA (TEMP2),Y
0068  7F55  C8              INY
0069  7F56  A9 FF           LDA #$FF
0070  7F58  91 89           STA (TEMP2),Y   ;OLD STRING DEAD
0071  7F5A  60       NOKILL RTS
0072  7F5B           ;
0073  7F5B  20 0B 7F FINDI  JSR CHKPAR      ;EVALUATE NUMERIC
0074  7F5E  20 2D C9        JSR FLTINT
0075  7F61  A5 12           LDA INTEG+1
0076  7F63  F0 03           BEQ R1
0077  7F65  4C 00 BF        JMP SYNERR
0078  7F68  A5 11     R1    LDA INTEG
0079  7F6A  60              RTS
0080  7F6B           ;
0081  7F6B           ;**         BLOCK GET ROUTINE       **
0082  7F6B  20 5B 7F BLKGET JSR FINDI        ;GET FILE NUMBER
0083  7F6E  85 D2           STA CURFIL
0084  7F70  A9 00           LDA #<BUFFER     ;GET BUFFER ADDRESS
0085  7F72  85 01           STA TEMP1
0086  7F74  A9 7E           LDA #>BUFFER
0087  7F76  85 02           STA TEMP1+1
0088  7F78  20 25 7F        JSR FINDB        ;FIND STRING VARIABLE
0089  7F7B  20 37 7F        JSR KILSTR       ;KILL OLD STRING
0090  7F7E  A6 D2           LDX CURFIL
0091  7F80  20 C6 FF        JSR $FFC6        ;SET INPUT DEVICE
0092  7F83  20 E4 FF INLOOP JSR $FFE4        ;GET A CHARACTER
0093  7F86  AA              TAX              ;SAVE CHAR IN .X
0094  7F87  C9 0D           CMP #$0D         ;CARRIAGE RETURN?
0095  7F89  F0 15           BEQ VARSET       ;YES, STOP INPUT
0096  7F8B  A0 00           LDY #$00
0097  7F8D  91 01           STA (TEMP1),Y    ;STORE CHAR IN BUFFER
0098  7F8F  E6 01           INC TEMP1        ;INCREMENT LENGTH
0099  7F91  A4 01           LDY TEMP1        ;GET LENGTH
0100  7F93  C0 FF           CPY #255         ;MAX ?
0101  7F95  F0 04           BEQ PRESET
0102  7F97  A4 96           LDY STATUS       ;MORE CHARS?
0103  7F99  F0 E8           BEQ INLOOP       ;YES, CONTINUE
0104  7F9B  8A       PRESET TXA              ;GET CHAR FROM .X
0105  7F9C  D0 02           BNE VARSET       ;LAST CHAR CHR$(0)?
0106  7F9E  C6 01           DEC TEMP1        ;YES, DEC LENGTH
0107  7FA0  20 CC FF VARSET JSR $FFCC        ;CLOSE CHANNEL
0108  7FA3  A0 00           LDY #$00
0109  7FA5  A5 01           LDA TEMP1
0110  7FA7  91 44           STA (VARPTR),Y   ;STOR LEN IN POINTER
```

```
NE# LOC   CODE        LINE

.11  7FA9  C8                 INY
.12  7FAA  A9 00              LDA #<BUFFER
.13  7FAC  91 44              STA (VARPTR),Y   ;STORE POINTER LOW
.14  7FAE  C8                 INY
.15  7FAF  A9 7E              LDA #>BUFFER
.16  7FB1  91 44              STA (VARPTR),Y   ;STORE POINTER HI
.17  7FB3  60                 RTS
.18  7FB4              ;
.19  7FB4              ;**          INSTRING ROUTINE          **
.20  7FB4  20 14 7F    INSTRG JSR FINDAB       ;FIND BOTH VARIABLES
.21  7FB7  20 5B 7F           JSR FINDI        ;GET INSERT POSITION
.22  7FBA  C6 11              DEC INTEG
.23  7FBC  A5 89              LDA TEMP2
.24  7FBE  18                 CLC
.25  7FBF  65 11              ADC INTEG
126  7FC1  85 89              STA TEMP2
127  7FC3  90 02              BCC I1
128  7FC5  E6 8A              INC TEMP2+1
129  7FC7  A0 00       I1     LDY #$00
130  7FC9  B1 01       I2     LDA (TEMP1),Y
131  7FCB  91 89              STA (TEMP2),Y    ;TRANSFER BYTES
132  7FCD  C8                 INY
133  7FCE  C4 00              CPY LEN1         ;END OF INSERT
134  7FD0  D0 F7              BNE I2           ;NO, DO MORE
135  7FD2  60                 RTS
136  7FD3              ;
137  7FD3              ;**        POSITION SEARCH ROUTINE        **
138  7FD3  20 14 7F    POSTRG JSR FINDAB       ;FIND BOTH VARIABLES
139  7FD6  A9 00              LDA #$00
140  7FD8  AA                 TAX              ;X REG = 0
141  7FD9  85 B6              STA $B6          ;RESET POSITION
142  7FDB  A0 00       LOOPP  LDY #$00         ;ZEROIZE OFFSET
143  7FDD  B1 01       P1     LDA (TEMP1),Y    ;GET CHAR AT A$,Y
144  7FDF  D1 89              CMP (TEMP2),Y    ;SAME AS CHAR AT B$,Y ?
145  7FE1  D0 0C              BNE BUMP         ;NO, MOVE TO NEXT A$ CHAR
146  7FE3  C8                 INY              ;YES, INCREMENT Y
147  7FE4  C4 00              CPY LEN1         ;SAME AS LEN OF A$?
148  7FE6  D0 F5              BNE P1           ;NO, MORE CHARS TO COMPARE
149  7FE8  E6 B6              INC $B6          ;YES, BUMP POSITION
150  7FEA  A5 B6              LDA $B6          ;AND STORE IT
151  7FEC  85 00              STA $00          ;IN LOCATION 0
152  7FEE  60                 RTS
153  7FEF  E6 89       BUMP   INC TEMP2        ;MOVE TO NEXT CHAR IN B$
154  7FF1  D0 02              BNE P2
155  7FF3  E6 8A              INC TEMP2+1
156  7FF5  E6 B6       P2     INC $B6          ;BUMP POSITION
157  7FF7  A5 B6              LDA $B6
158  7FF9  C5 88              CMP LEN2         ;END OF B$ ?
159  7FFB  D0 DE              BNE LOOPP        ;NO, DO MORE COMPARES
160  7FFD  86 00              STX $00          ;NOT FOUND
161  7FFF  60                 RTS
162  8000                     .END

RRORS = 0000
```

## Machine Code TO DATA Statements

This small program can be used to take bytes out of memory and put them into DATA statements. The program is self modifying. Once complete, the program itself must be removed from text, leaving the DATA statements behind.

Line 3 prompts for the starting DATA line number. This should obviously be out of the line range of the program. Line 4 prompts for how many bytes per DATA statement. The number of bytes extracted from memory will be the difference of the starting line number and the ending line number.

This program is an adaptation from one written by Dave Middleton, Commodore U.K.

```
1 INPUT " STARTING ADDRESS IN DECIMAL ";AD
2 INPUT "   ENDING ADDRESS IN DECIMAL ";EN
3 INPUT "          STARTING LINE NUMBER  1000[CL CL CL CL CL CL]";LN
4 INPUT "# OF BYTES/DATA LINE (MAX 13)  8[CL CL CL]";NB
5 PRINT"[CLR]"LN;"FORJ="AD;"TO"EN;":READ X:POKE J,X:NEXT:END":J=NB:GOTO16
6 IF AD>EN THEN PRINT"[CLR]DONE.  NOW ERASE GENERATOR" : END
7 FOR J=1 TO NB
8 V=PEEK(AD) : AD=AD+1
9 S$=STR$(V)
10 A$=A$+RIGHT$("  "+S$,4)
11 IF AD>EN THEN LN=LN-NB+J : GOTO14
12 IF J<NB THEN A$=A$+","
13 NEXT J : J=J-1
14 A$=STR$(LN)+" DATA"+A$
15 PRINT"[CLR]"A$
16 PRINT"AD="AD;":EN="EN;":LN="LN;"+"J;":NB="NB;":GOTO6"
17 POKE 158, 3 : POKE 623,19 : POKE 624,13 : POKE 625,13 : END
```

## PET BASIC Label Support Interface; Revisited

In Transactor #12, Volume 2, J. Hoogstraat of Calgary submitted an excellent routine which allows alpha labelling of BASIC statements. This routine is most useful when developing software. By labelling your subroutines with words, a renumber operation doesn't force you to remember a whole new set of line numbers. When developement is complete, the labels can be swapped back to line numbers using a "find/change" function like in 'The Toolkit' or 'AID4'.

The routine published last issue was for BASIC 2.0. Here are two new versions for BASIC 4.0; one for use with cassette and the other for use with disk.

Recall that BASIC 4.0 disk commands (eg. DOPEN, BACKUP, SCRATCH, etc.) use parts of the 2nd cassette buffer which would clobber any machine language set up there previously. Use the version that resides in the 1st cassette buffer (634 to 816) if any of these disk commands are coded elsewhere in your program.

On the other hand, cassette users will probably want to use cassette port #1 which prohibits using cassette buffer #1 for anything else. In that case, use the version that resides in the 2nd cassette buffer (826 to 1008). It will be safe there so long as no 4.0 disk commands get executed.

The version you choose will determine the SYS address used to engage the routine into BASIC (see example)

```
10 SYS634 :REM USE SYS826 FOR 2ND CASSETTE BUFFER
20 :
100 FOR I=1 TO 3
110 ON I GOSUB #SUB1, #SUB2, #SUB3
120 NEXT
130 GOTO #ALLDONE
140 :
150 #SUB1:PRINT"SUBROUTINE";I : RETURN
510 :
550 #SUB2
560 PRINT"SUBROUTINE";I : RETURN
570 :
600 #SUB3 : PRINT"SUBROUT";I : RETURN
610 :
800 #ALLDONE : PRINT "END OF TEST" : END
```

```
600 FOR J=634 TO 816 : READ X :POKE J, X :NEXT
634 DATA 169, 135, 133, 113, 169,   2
640 DATA 133, 114, 169,  76, 133, 112
646 DATA  96, 230, 119, 208,   2, 230
652 DATA 120, 164,  55, 200, 208,   3
658 DATA  76, 118,   0, 160,   0, 177
664 DATA 119, 201,  35, 208, 245, 186
670 DATA 189,   1,   1, 201, 193, 240
676 DATA  24, 201,  47, 240,  20, 201
682 DATA  18, 240,  16, 201, 236, 208
688 DATA 107,  32, 112,   0, 201,  44
694 DATA 208, 249, 104, 104,  76, 226
700 DATA 184, 200, 166,  40, 165,  41
706 DATA 208,   8, 160,   0, 177,  92
712 DATA 170, 200, 177,  92, 134,  92
718 DATA 133,  93, 133,  91, 177,  92
724 DATA 208,   3,  76, 110, 184,  24
730 DATA 165,  92, 105,   4, 133,  90
736 DATA 144,   2, 230,  91, 136, 177
742 DATA  90,  32,  34,   3, 133,  89
748 DATA 177, 119, 200,  32,  34,   3
754 DATA 197,  89, 208, 206, 201,   0
760 DATA 208, 235, 104, 104, 186, 189
766 DATA 255,   0, 201,  18, 208,  21
772 DATA 165, 120,  72, 165, 119,  72
778 DATA 165,  55,  72, 165,  54,  72
784 DATA 169, 141,  72, 169, 183,  72
790 DATA 169,  73,  72,  32,  80, 184
796 DATA  32, 131, 184,  76, 118,   0
802 DATA 201,  32, 240,   8, 201,  58
808 DATA 240,   4, 201,  44, 208,   2
814 DATA 169,   0,  96
```

```
820 FOR J=826 TO 1008 : READ X :POKE J, X :NEXT
826 DATA 169,  71, 133, 113, 169,   3
832 DATA 133, 114, 169,  76, 133, 112
838 DATA  96, 230, 119, 208,   2, 230
844 DATA 120, 164,  55, 200, 208,   3
850 DATA  76, 118,   0, 160,   0, 177
856 DATA 119, 201,  35, 208, 245, 186
862 DATA 189,   1,   1, 201, 193, 240
868 DATA  24, 201,  47, 240,  20, 201
874 DATA  18, 240,  16, 201, 236, 208
880 DATA 107,  32, 112,   0, 201,  44
886 DATA 208, 249, 104, 104,  76, 226
892 DATA 184, 200, 166,  40, 165,  41
898 DATA 208,   8, 160,   0, 177,  92
904 DATA 170, 200, 177,  92, 134,  92
910 DATA 133,  93, 133,  91, 177,  92
916 DATA 208,   3,  76, 110, 184,  24
922 DATA 165,  92, 105,   4, 133,  90
928 DATA 144,   2, 230,  91, 136, 177
934 DATA  90,  32, 226,   3, 133,  89
940 DATA 177, 119, 200,  32, 226,   3
946 DATA 197,  89, 208, 206, 201,   0
952 DATA 208, 235, 104, 104, 186, 189
958 DATA 255,   0, 201,  18, 208,  21
964 DATA 165, 120,  72, 165, 119,  72
970 DATA 165,  55,  72, 165,  54,  72
976 DATA 169, 141,  72, 169, 183,  72
982 DATA 169,  73,  72,  32,  80, 184
988 DATA  32, 131, 184,  76, 118,   0
994 DATA 201,  32, 240,   8, 201,  58
1000 DATA240,   4, 201,  44, 208,   2
1006 DATA169,   0,  96
```

## The Commodore 8010 Modem

Gord Campbell, Toronto

Hurray! It works! In fact, the main benefit of the Commodore Modem (CBM 8010) is that it is so easy to use. You just take it out of the box, plug it in, attach an IEEE 488 cable, and you have completed the installation.

But what does it do? It allows the PET or CBM to communicate with other 'similar' devices. Thus you can communicate with not only another PET with a modem, but also with mainframe computers or a wide variety of terminals. The only restriction is that the other end must also run at 300 baud (30 characters per second), and be 'BELL-103' compatible. Since BELL-103 is the North-American standard, this is not a restriction. Most terminals these days support 300 baud: the major exception is the good OLD Teletype.

To establish communications, one end phones the other, agreement is made on who will run in answer mode, and both ends place their telephone handset in the cups in the top of the modem. The modems sing away at each other, sending and receiving characters.

Like any device on the PET, it doesn't do anything without a program, but that isn't a big deal. In the 11 page manual are listings of programs to do PET-to-PET or Pet-to-mainframe communications, as well as a program which helps to pin down any problems which might occur. Unfortunately, at least one of these has a bug (take line 240 out of the terminal program). Since many of the people who might like to use the PET as a terminal have no interest in programming, it would have been nice to have a tape right in the box. Presumably any dealer who is selling modems has working versions of the programs.

The most difficult concept to grasp in programming for the modem is that the program must handle events which are coming from an external source, possibly quite quickly. For example, a mainframe computer will send characters to you at the rate of 30 per second. If you fall behind in processing them, you miss a few. A BASIC program to accept characters from the modem and write them to disk must be structured for speed or it will lose some. This will be less of a problem if the other end is producing characters at someone's typing speed. Any extra overhead which slows down programs (such as DOS support) will make the problem worse of course.

The commands used in programming the modem are the same as for other devices (OPEN, CLOSE, PRINT#, GET#, and INPUT#). Thus it should be simple to set up two-player games by phone. One of my desires is a program to play bridge by phone: me and the PET versus a friend and his computer. The computers would get to do all the dull stuff like deal, keep score, and be dummy. The only clever things they would have to do is bid and play defense. The easy part of such a program would be the part handling the modem.

The price of the modem does seem a little steep (about $600), but this is a professional, well-integrated peripheral. The owners of the other two major personal computer systems have to pay about the same amount to obtain the officially-approved equivalent capability. The only reason that it seems so much is that there are other ways to get communications without paying as much. Unfortunately,the other ways are not nearly so easy to use.

Are there any complaints? A couple of minor ones, but no big deal. The Commodore modem does not allow sending a true 'break'. Many large computers accept a break (which is not a character as such) to stop them from doing what they are doing. For example, if you are editing a large file using IBM's TSO (time sharing option), and you accidentally say LIST, you will have to wait a long time before you can enter another command. (This can be overcome by telling TSO, 1. don't send more than say 20 lines without giving me a shot, and 2. if I send you a certain character string, take that to be a 'break').

Finally, the main thing which makes the modem so easy to use (the IEEE bus), can be a drawback at times. For example, it is very difficult to use the modem and the printer at the same time. This is because the printer will not accept the first character of a line until it has finished printing the previous line. By the time your program continues, you have lost characters. This could have been avoided by giving the modem a fair sized buffer, but that would have made the price even higher.

In summary, if you want to add data communications to the PET, the Commodore modem deserves consideration. For hassle-free installation and operation, it is probably your best choice!

Editor's Note

Two excellent machine language modem drivers have been sent to all Canadian Commodore dealers (CEAB #6 disk). These are free for copying to any customer but they will only work the 8010 modem.

## Spooling Disk Files to Printers

In Compute #8 , T.M. Peterson published a neat trick for getting the 2040 disk to talk to a printer without PET/CBM supervision of the IEEE bus. I imagine this would work for 4040s, 8050s and any make printer interfaced via the IEEE bus, but naturally I can't be sure for all cases. However, the idea was so incredible that I felt it definitely worth repeating.

Everyone knows how to LIST a program to the printer. But long listings can wear patience thin, especially on a slow printer! Not only that, but while your printer is chugging along, the PET just sits there with everthing disabled except RUN/STOP. Wouldn't it be nice if the disk fed the printer while you continue editing or play a quick round of space invaders or Microchess, that is if you can bear some of those arrogant printers. By the way, those wing nuts on the bottom of Commodore 202X printers... take them out. They're only shipping screws that hold the mechanism tight. Once removed, the noise level is reduced considerably.

First a file must be created on disk. This could be any SEQ file with any contents that are printer recognizable, but for now we'll create one of a program LISTing.

```
1.  Enter some small program
2.  OPEN 8,8,8,"0:TEST SPOOL,S,W" : CMD8 : LIST
3.  PRINT#8,""; : CLOSE8
4.  NEW
5.  OPEN 8,8,8,"0:TEST SPOOL"       ;defaults to ',s,r'
6.  POKE 165,72  : SYS 61695        ;use SYS 61668
7.  POKE 165,104 : SYS 61695          for BASIC 2
8.  OPEN 4,4 : CMD4 : POKE 176,3 : POKE 174,0
```

On hitting return the printer should fire up and continue at full speed to the end of the file. At this point your cursor might be acting funny (try hitting return on a blank line). To stop this, POKE 14,0 for BASIC 2.0 or POKE 16,0 for BASIC 4.0. If you're lazy like me, invoking a couple of ?SYNTAX ERRORS (eg. '=' and Return) will do the same thing. However, to restore normal cursor operation under program control (yes program control!), you would have to use the POKE. More on this in upcoming paragraphs.

Once the printer starts, don't try using the IEEE bus or the spool will abort. When its all finished you can CLOSE the open disk file by sending an Initialize command or with:

```
OPEN 1,8,8 : CLOSE 1
```

A filename isn't necessary, but use the same secondary address as in step 5.

The NEW command at step 4 is only for clarity. Instead you might load another program or just leave the current one in for further editing. You can RUN the program in memory and even use the cassettes, but they're still as slow as before.

The example here uses all direct commands but they could just as easily be put in a program. Think of the applications! In a user oriented system, a report could be output to the disk and immediately spooled to the printer while the operator continues working on the next task. Of course the user might inadvertently try a bus operation which would kill everything. Fortunately this busy state can be detected using the following "trap":

```
100 IF (NOT(PEEK(59456))) AND 64 THEN 100
110 OPEN 1,8,SA : CLOSE 1
130 ...and continue
```

If a spool is in progress, line 100 will loop back to itself until the bus is free. Line 110 is for closing the disk file and also turns off the active LED. SA is a variable containing the secondary address which might be used in coding the OPEN command that starts the operation. Also note that line 110 causes no disk activity so there's no need to go around it to save time.

Theory and Variations

This example uses device number 8 right through. However, you might have more than one disk on line which would mean a different device number. In step 6, address 165 (the IEEE output buffer) is POKEd with 72. This number (72) is derived from 64 + 8, where 8 is the device number. For versatility, this '8' might be replaced by a variable like DV.

The following SYS activates the ATN line on the bus telling all devices to 'pay attention'. The contents of 165 are then sent to the bus but only the device that has a matching 'talk' address responds, in this case device 8; the disk.

The disk is ready to start sending but from where? All it needs now is the secondary address of the OPENed file. Step 7 sets up the output buffer with the secondary address plus 96. Since 8 was chosen, the result is 104. This step might also be modified to read POKE 165,96+SA.

Nothing happens yet because the ATN line isn't released by the PET. When CMD4 is executed (step 8), the printer becomes the output command device. PET releases ATN, the disk starts talking and the printer listens.

POKE 176,3 tricks the PET into thinking the output command device is the screen and POKE 174,0 simulates no files OPEN. In a program you would have to re-open files (eg. command channel, modem, etc.) at spool completion. By the same token, you might want to CLOSE any open write files before starting.

Should anyone discover any useful variations to this technique, let us know. We'll be glad to here about it!

POKE OPEN Files

If a file is OPENed from a program and a ?SYNTAX ERROR occurs before the program gets a chance to CLOSE it, you can still CLOSE it from the keyboard using a direct command (i.e. CLOSE 1f). But if you edit the program before CLOSing, you'll find this is no longer possible.

Editing essentialy does a CLR which also aborts all file activity. However, nothing is sent to the disk to close its' open files. If these were 'read' files, send an Initialize or Catalog command and the disk is back to normal.

OPEN 'write' files are a different story. If they aren't CLOSEd properly, a number of nasty conditions can occur that are no fault of the DOS.

All disk files (PRG, SEQ, REL or USR) are created by recording data in sectors around the disk and then linking them together using two bytes to store the track and sector co-ordinates of the next sector in the chain. The first link is stored in the directory as soon as the file is OPENed for write. As successice sectors are written, successive links are also recorded (first two bytes of sector). Thus the directory points at the first sector, the first sector points at the second, and so on.

When the DOS closes a write file, an 'end marker' is recorded in the last sector. This is characterized by setting the track byte to zero (since there is no track 0), and pointing the sector byte at the last character recorded. If the file is not closed, this end marker never gets stored. An "open chain" condition results.

Blocks that haven't been used contain all zeroes. The previous link would point at two zeroes which is ok. However, blocks that have been used and then freed due to some earlier scratch operation, still contain old links. If an open chain points at one of these, a "phantom chain" is created. This can be deadly!

Consider a Scratch operation on an improperly closed file. The DOS would follow the chain, releasing sectors to the BAM (Block Availability Map) right up to the point where the write was aborted. But if a phantom chain exists, the DOS continues along the chain releasing more sectors which might lead right into a good file! In the next write operation, the DOS might select one of these sectors and clobber live data (BLECCH!).

Fortunately we have the Validate or Collect command. A Collect will discard improperly closed files, freeing all sectors that do not belong to a chain and allocating all sectors that do. Collect also frees any allocated direct access blocks, which may not be too desirable. Besides that, an 8050 Collect operation can take awfully long if your

directory contains many entries. Wouldn't it be nice if we could just CLOSE that pesky open file, scratch it, and start all over?!

Well,... you can! As long as the disk is not disturbed by a Catalog, reset, etc. When the PET aborts external file activity due to editing, a CLR, etc., it merely sets the number of OPEN files to zero (address 174). All file parameter tables are left in tact (addresses 593 through 622). These files can be ressurected as long as no new files have been OPENed.

POKE 174, X

... where 'X' is the number of OPEN files. If you don't know how many files were OPEN, simply POKE 174,10. This sets it to maximum at which point even some properly CLOSEd files would now be OPEN again! Now you can issue any necessary CLOSE commands directly from the keyboard and exterminate those disk gremlins that show up as an asterisk beside the file type.

Although this will get you out of a jam, it is not suggested practice! CLOSing disk files properly will reduce knashing-of-teeth and pulling-of-hair considerably! (right Jim?)

Some Commodore Disk Utilties

The following routines are for use with 4040 and 8050 disk units.

The first two are for reading the disk ID from the specified drive DR, device DV. Line 150 sets the drive. Line 160 sets the track. Line 170 tells the disk to read any header on track 18. Initializing not necessary! 180, 190 and 200 wait for the DOS to finish the read. Line 210 does any error processing (i.e. read error or no disk in drive). The first character of the ID is read from DOS memory (line 220) and the DOS puts it in the command channel. Line 250 reads the second character and both characters are put in 'ID$'.

This routine is particularly usetul (and fast) to see if the user has placed a disk in the drive. It can also be used to detect insertion of incorrect diskettes. The software would have to anticipate ID numbers, perhaps ID's that were selected by the program in an earlier formatting operation.

The second two will return the BLOCKS FREE count BF, from the specified drive DR, device DV. BF is reset before entering. Initialize is necessary here! The block free count is not stored on the disk but rather calculated from the Block Availibilty Map. Line 170 sends the DOS oft to that routine in disk ROM! The result is placed in disk RAM where it is read, once again, into the command channel by lines 180 and 210. A few calculations and presto! Block Free!

Knowing blocks-free from within a program can be especially useful for anticipating the nasty DISK FULL error.

Versions for both 4040 (DOS 2.0) and 8050 (DOS 2.5) have been provided. The fifth program was written by John Collins of the U.K. It's the definitive subroutine for determining host equipment.

```
100 REM ID READER FOR 4040
110 ID$="" : REM RESET ID$
120 DR=0   : REM DRIVE #
130 DV=8   : REM DEVICE#
140 OPEN15,DV,15 : REM UNLESS ALREADY OPEN
150 PRINT#15,"M-W"CHR$(18) CHR$(0) CHR$(1)CHR$(DR)
160 PRINT#15,"M-W"CHR$(43) CHR$(16)CHR$(1)CHR$(18)
170 PRINT#15,"M-W"CHR$(4)  CHR$(16)CHR$(1)CHR$(176+DR)
180 PRINT#15,"M-R"CHR$(4)  CHR$(16)
190 GET#15,X$
200 IF ASC(X$)>127 THEN 180
210 IF ASC(X$)<>1 THEN PRINT#15,"M-E"CHR$(37)CHR$(217):PRINTDS,DS$:END
220 PRINT#15,"M-R"CHR$(41)CHR$(16)
230 GET#15,A$
240 ID$=A$
250 PRINT#15,"M-R"CHR$(42)CHR$(16)
260 GET#15,A$
270 ID$=ID$+A$
280 PRINTID$
```

```
100 REM ID READER FOR 8050
110 ID$=""          : REM RESET ID$
120 DR=0            : REM DRIVE #
130 DV=8            : REM DEVICE#
140 OPEN15,DV,15 : REM UNLESS ALREADY OPEN
150 PRINT#15,"M-W"CHR$(18) CHR$(0) CHR$(1)CHR$(DR)
160 PRINT#15,"M-W"CHR$(43) CHR$(16)CHR$(1)CHR$(18)
170 PRINT#15,"M-W"CHR$(4)  CHR$(16)CHR$(1)CHR$(176+DR)
180 PRINT#15,"M-R"CHR$(4)  CHR$(16)
190 GET#15,X$
200 IF ASC(X$)>127 THEN 180
210 IF ASC(X$)<>1 THEN PRINT#15,"M-E"CHR$(179)CHR$(238):PRINTDS,DS$:END
220 PRINT#15,"M-R"CHR$(41)CHR$(16)
230 GET#15,A$
240 ID$=A$
250 PRINT#15,"M-R"CHR$(42)CHR$(16)
260 GET#15,A$
270 ID$=ID$+A$
280 PRINTID$
```

```
100 REM SUBROUT. RETURNS BLOCKS-FREE FOR DOS 2.0
110 DR=0 : REM DR = DRIVE #
120 DV=8 : REM DV = DEVICE#
130 BF=0 : REM RESET BLOCK FREE COUNT
140 OPEN15,DV,15          : REM UNLESS ALREADY OPEN
150 PRINT#15,"I"+STR$(DR): REM UNLESS ALREADY INIT'D
160 PRINT#15,"M-W"CHR$(18)CHR$(0)CHR$(1)CHR$(DR)
170 PRINT#15,"M-E"CHR$(52)CHR$(219)
180 PRINT#15,"M-R"CHR$(119)CHR$(67)
190 GET#15,A$
200 BF=ASC(A$+CHR$(0)) : REM IN CASE A$=""
210 PRINT#15,"M-R"CHR$(120)CHR$(67)
220 GET#15,A$
230 BF=BF+ASC(A$+CHR$(0))*256
240 PRINT BF
```

```
100 REM SUBROUT. RETURNS BLOCKS-FREE FOR DOS 2.5 (8050)
110 DR=0 : REM DR = DRIVE #
120 DV=9 : REM DV = DEVICE#
130 BF=0 : REM RESET BLOCK FREE COUNT
140 OPEN15,DV,15          : REM UNLESS ALREADY OPEN
150 REMPRINT#15,"I0"
160 PRINT#15,"M-W"CHR$(18)CHR$(0)CHR$(1)CHR$(DR)
170 PRINT#15,"M-E"CHR$(231)CHR$(211)
180 PRINT#15,"M-R"CHR$(158)CHR$(67)
190 GET#15,A$
200 BF=ASC(A$+CHR$(0)) : REM IN CASE A$=""
210 PRINT#15,"M-R"CHR$(160)CHR$(67)
220 GET#15,A$
230 BF=BF+ASC(A$+CHR$(0))*256
240 PRINT BF
```

```
100 REM VERSION TEST FOR PET/CBM AND DISK
110 REM BY:  JOHN COLLINS
120 REM PET/CBM TEST  TP;
130 REM 2001 BASIC 1.0 = 0
140 REM 2001 BASIC 2.0 = 1
150 REM 4032 BASIC 4.0 = 2
160 REM 8032 BASIC 4.0 = 3
170 :
180 A=PEEK(57345):TP=0:IF A THEN TP=1:IF A AND 1 THEN TP=3:IF A AND 4 THEN TP=2
190 :
200 REM DISK TEST     TD;
210 REM 2040 (DOS 1.0) = 1
220 REM 4040 (DOS 2.0) = 2
230 REM 8050 (DOS 2.5) = 3
240 :
250 OPEN 15, 8, 15
260 PRINT#15,"M-R"CHR$(255)CHR$(255)
270 GET#15, A$
280 CLOSE 15
290 A=ASC(A$):TD=1:IF A AND 16 THEN TD=3:IF A AND 1 THEN TD=2
300 :
310 REM RESULTS:
320 PRINT TP, TD
```

## Cursor Coding

Publishing programs is one main function of The Transactor. But publishing legible code is important too. Using the Spinwriter does this well but it won't print any of the special graphic characters that make their way into so many programs. The following method of representing graphics will be used in The Transactor:

1. Square brackets ("[CL]") within quotes indicates cursor graphics.
2. Where necessary, successive apostraphes (') will be enclosed in square brackets to represent successive spaces.
3. Any other graphics will be converted to their CHR$ equivalents to accomodate business keyboard users.
4. Special 8032 screen operators will also be represented by CHR$.

| | | |
|---|---|---|
| "[CLR]" | = | clear screen |
| "[HOME]" | = | cursor home |
| "[UP]" | = | cursor up |
| "[DN]" | = | cursor down |
| "[CL]" | = | cursor left |
| "[CR]" | = | cursor right |
| "[RVS]" | = | reverse mode on |
| "[OFF]" | = | reverse mode off |
| "[ ' ]" | = | 1 space |

## Examples

| | | |
|---|---|---|
| "[CLR DN DN DN]" | = | clear screen followed by 3 cursor downs, no spaces. |
| "[CL CL CL]" | = | three cursor lefts, no spaces |
| "[RVS ''''' OFF]" | = | reverse mode on, 5 spaces, reverse oft |
| "[CL ']" | = | 1 cursor left, 1 space |

Tax Ontario 1980

Here it is ! Jim Butterfields income tax return program for 1980. If you still have one of Jims earlier tax programs, pull it out and you may avoid a lot of unecessary typing. The program will work on any BASIC, 8K minimum. It also calculates your tax refund (or payment as the case may be) to the penny! No unfair rounding the way the tax tables in the book do it.

```
100 DIMC(5),F(5),D(13):REM V3.0 APR3/81
110 S$="[''''''''''''''''''''''''''''''''''''''''''''''''''']"
120 PRINT"[CLR DN]ONTARIO INCOME TAX 1980 TAXATION YEAR"
130 PRINT"['' RVS]J BUTTERFIELD"
140 FORJ=1TO90:IFPEEK(J+32768)=32THENNEXTJ
150 L=J
160 INPUT"[DN]INSTRUCTIONS";Z$:IFASC(Z$)=78GOTO240
170 PRINT"[DN]ONTARIO INCOME TAX FOR 1980"
180 PRINT"[DN]FOLLOW YOUR FORM: THIS PROGRAM WILL"
190 PRINT"HELP WITH THE ARITHMETIC."
200 PRINT"[DN]FOR 'NIL' ITEMS, JUST PRESS 'RETURN'."
210 PRINT"[DN]FOR 'MULTIPLE' ENTRIES, ENTER AMOUNT"
220 PRINT"AND PRESS '+' INSTEAD OF 'RETURN' TO"
230 PRINT"SIGNAL MORE ITEMS TO COME.[DN]"
240 INPUT"PRINTER(Y/N)";Z$:IFASC(Z$)=89THENOPEN2,4:P5=-1
250 DEFFNS(M)=(M+S-ABS(M-S))/2
260 DEFFNB(M)=(M+B+ABS(M-B))/2
270 DEFFNP(M)=INT(M*P/100+.49)
280 DEFFNI(M)=INT(M*100+.5)
290 GOSUB1340
300 P1=1:GOSUB1440
310 I$="INCOME FROM EMPLOYMENT":GOSUB1490
320 P=3:S=5E4:GOSUB1380:I=FNS(FNP(I)):I$="LESS EMPLOYMT EXPENS":GOSUB1400
330 I$="*NET EMPL EARNINGS":GOSUB1420
340 I$="DIVD FROM CDN CORP":GOSUB1510
350 D(0)=I:I$="INTEREST & INV INCM":GOSUB1510
360 D(1)=I:I$="CANADIAN CAPITAL GAINS":GOSUB1510
370 D(4)=I:I$="ALL OTHER INCOME":GOSUB1510
380 I$="**TOTAL INCOME**":GOSUB1420
390 D(2)=I:P1=2:GOSUB1440:GOSUB1720
400 PRINT"LESS:":GOSUB1380:I$="CPP CONTRIBUTIONS":GOSUB1490
410 I$="UIC PREMIUMS":GOSUB1510
420 I$="OTHER DEDUCTIONS":GOSUB1510
430 I$=" *TOT DEDUCTIONS:":GOSUB1420
440 I=C(C):I$=" *NET INCOME*":D(3)=I:GOSUB1720
450 PRINT"[RVS]EXEMPTIONS:":GOSUB1380
460 I=289E3:I$="BASIC EXEMPTION":GOSUB1680
470 I$="AGE EXEMPT":GOSUB1510:I$="MARRIED EXEMPT":GOSUB1510
480 I$="DEPNDT CHILD EXMPT":GOSUB1510:I$="OTHER EXMPT":GOSUB1510
490 I$=" *TOTAL EXEMPT*":GOSUB1420:D(12)=I
500 I$=" **LINE 46**":GOSUB1660
510 GOSUB1380:GOSUB1360:I$="MEDICAL EXPENSES":GOSUB1490
520 IFI=0GOTO550
530 GOSUB1380:P=3:I=FNP(D(3)):I$="*LESS 3% N.I.":GOSUB1400
540 I$="ALLOWABLE MED EXP":GOSUB1660
550 I$="CHARITABLE DONATNS":GOSUB1510:I=C(C)
```

```
0 C=C-1:B=1E4:I=FNB(I):I$="**STANDARD DEDCTN**":GOSUB1680
0 S=1E5:I=FNS(D(0)+D(1)+D(4)):D(10)=I
0 IFI>0THENI$="*I, D & CG DEDUCTION":GOSUB1700
0 I$="ALL OTHER DEDUCTIONS":GOSUB1510
0 I$="**TOTAL DEDUCTIONS":GOSUB1420
0 I$=" **TAXABLE INCOME**":GOSUB1660:D(5)=I:GOSUB1840
0 IFI<=30E5ANDD(0)=0THENPRINT"YOU MAY USE TAX TABLE .. OR..."
0 P1=1:GOSUB1460
0 DATA108360,35849,43
0 DATA70434,21058,39
0 DATA43344,11306,36
0 DATA25284,5526,32
0 DATA19866,4009,28
0 DATA16254,3106,25
0 DATA12642,2276,23
0 DATA9030,1517,21
0 DATA5418,831,19
0 DATA3612,506,18
0 DATA1806,199,17
0 DATA903,54,16
0 DATA0,0,6
0 DATA-1
0 READX,Y,P:IFI<X*100GOTO780
0 T=Y*100:P$="ON FIRST $"+STR$(X)+" TAX IS      "+STR$(Y):GOSUB1820
0 J=I-X*100:I=FNP(J)
0 P$="ON RMG $"+STR$(J/100)+" TAX AT"+STR$(P)+"% IS $"+STR$(I/100)
0 GOSUB1820
0 I=I+T:GOSUB1340:C=C-1:I$="TOTAL FED INCM TAX":GOSUB1680
0 S=I:P=25:I=FNS(FNP(D(0))):D(11)=I
0 IFI>0THENGOSUB1380:I$="DIV TAX CREDIT":GOSUB1400
0 I$=" *BASIC FEDERAL TAX*":I=C(C):GOSUB1720:GOSUB1840:D(6)=I
0 IFI<2E4GOTO900
0 IFI<=2222E2THENI=2E4:GOTO900
0 P=9:S=5E4:I=FNS(FNP(I))
0 GOSUB1380:I$="GENERAL TAX REDUCTION":GOSUB1400
0 I$=" **FEDERAL TAX**":GOSUB1660:D(7)=I
0 GOSUB1380:I$="FOREIGN TAX PAID":GOSUB1510:IFI=0GOTO980
0 W=I:I$="FORGN INCOME":GOSUB1510:K=I:X=(D(3)-D(10))/100:Y=(D(7)+D(11))/100
0 S=INT(K/X*Y)
0 P$=STR$(K/100)+"/"+STR$(X)+"*"+STR$(Y)+" ="+STR$(S/100):GOSUB1820
0 I=FNS(W):I$="--DEDUCT:":GOSUB1400
0 PRINT"..ANOTHER COUNTRY...":GOTO920
0 C=C-1:I$="FEDERAL TAX PAYABLE":GOSUB1420:D(8)=I:GOSUB1840
0 D1=D(6):IFD(5)<182E3THEND1=0
0 P=44:I=FNP(D1)
0 I$="BASIC ONTARIO TAX":GOSUB1720:D(9)=I
0 READX:IFX<>-1GOTO1020
0 P$=" ==ONTARIO PROPERTY & SALES TAX==":GOSUB1820:GOSUB1340
0 INPUT"ARE YOU ELIGIBLE FOR THESE CREDITS  Y[CL CL CL]";Z$
0 IFASC(Z$)<>89GOTO1170
0 I$="TOTAL RENT PAYMENTS":GOSUB1490:IFI=0THENC=C-1:GOTO1080
0 P=20:I=FNP(I):I$="*20% OF RENT":GOSUB1400
0 I$="PROPERTY TAXES&COLLG RES":GOSUB1510
0 I=C(C):P=10:X=FNP(I):I$="*OCCUPANCY COST*":GOSUB1420:GOSUB1840:C(C+1)=0
```

| | | |
|---|---|---|
| "[CLR]" | = | clear screen |
| "[HOME]" | = | cursor home |
| "[UP]" | = | cursor up |
| "[DN]" | = | cursor down |
| "[CL]" | = | cursor left |
| "[CR]" | = | cursor right |
| "[RVS]" | = | reverse mode on |
| "[OFF]" | = | reverse mode off |
| "[ ' ]" | = | 1 space |

```
1100 GOSUB1340
1110 S=18E3:I=FNS(I):I$=" ADD..":GOSUB1680:I=X:I$=" TO..":GOSUB1700
1120 I$="PROPERTY TAX CREDIT":GOSUB1420
1130 P=1:I=FNP(D(12)):I$="SALES TAX CREDIT":GOSUB1700
1140 I$="TOTAL CREDITS":GOSUB1420
1150 D1=D(5):IFD1<182E3THEND1=0
1160 GOSUB1380:P=2:I=FNP(D1):I$="LESS(B)--":GOSUB1400
1170 B=0:S=5E4:I=FNS(FNB(C(C))):I$="ONTARIO P & S CREDITS":GOSUB1400
1180 GOSUB1360:I$="ONT POLITICAL TAX CREDIT":GOSUB1490
1190 I$="*TOTAL ONT TAX CREDITS":I=C(C-1)+I:D(13)=I:GOSUB1400
1200 GOSUB1340
1210 P1=4:GOSUB1440:GOSUB1340:I=D(8):I$="FEDERAL TAX PAYABLE":GOSUB1400
1220 I$="POLIT/BUS/EMPLMT CREDIT":GOSUB1510:X=D(8)+D(9)-I
1230 C=C-1:I$="ONTARIO TAX PAYABLE":I=D(9):GOSUB1720
1240 C=C-1:I$="TOTAL PAYABLE":I=X:GOSUB1720:GOSUB1840
1250 GOSUB1380:I$="TAX DEDUCTED PER SLIPS":GOSUB1490
1260 I$="ONTARIO TAX CREDITS":I=D(13):GOSUB1700
1270 I$="OVERPAYMENTS/INSTALMENTS":GOSUB1510
1280 I$="CHILD TAX CREDIT":GOSUB1510
1290 I=C(C):C=3:I$="**TOTAL CREDITS**":GOSUB1720:GOSUB1840
1300 C=2:I$="BALANCE DUE":I=X-I:IFI<0THENI$="REFUND:":I=ABS(I)
1310 GOSUB1720:PRINT:IFP5THENFORJ=1TO10:PRINT#2:NEXTJ:CLOSE2
1320 END
1330 REM CLEAR ALL ACCUMS
1340 C=1:C(1)=0:GOSUB1360:GOSUB1360
1350 REM MOVE TO SUBTOTAL
1360 C=C+1:F(C)=1:C(C)=0:RETURN
1370 REM MOVE TO NEG SUBTOTAL
1380 C=C+1:F(C)=-1:C(C)=0:RETURN
1390 REM SUM I INTO NEXT HIGHER TOTAL
1400 C(C)=I
1410 REM SUM C(C) INTO NEXT HIGHER TOT
1420 I=C(C):F=F(C):C=C-1:C(C)=C(C)+I*F:GOTO1720
1430 REM PRINT PAGE ID
1440 P$="   ===PAGE":GOTO1470
1450 REM PRINT SCHED ID
1460 P$="   ===SCHEDULE"
1470 GOSUB1900:P$=STR$(P1)+" OF RETURN===":GOTO1820
1480 REM PROMPT NEW VALUE
1490 C(C)=0
1500 REM PROMPT INPUT
1510 I=0:GETZ$:PRINTI$;"? ";
1520 Y$="":PRINT CHR$(166)"[CL]";
1530 GETZ$:IFZ$=""GOTO1530
1540 Z=ASC(Z$):IFZ>47ANDZ<58GOTO1630
1550 IFZ$="-"ANDY$=""GOTO1630
1560 IFZ$="."GOTO1630
1570 IF(Z=157ORZ=20)ANDY$<>""THENY$=LEFT$(Y$,LEN(Y$)-1):PRINT"[CL ']";:GOTO1640
1580 IF Z$<>"+" THEN 1600
1590 PRINT"[DN]";:I=I+VAL(Y$):FORJ=1TOLEN(Y$):PRINT"[CL]";:NEXT:GOTO1520
1600 IFZ=13ANDI=0THENPRINT"[UP]";
1610 IFZ=13THENI=FNI(I+VAL(Y$)):PRINT:GOTO1700
1620 GOTO1530
1630 Y$=Y$+Z$
```

```
1640 PRINTZ$;:GOTO1530
1650 REM FORCE NON NEGATIVE
1660 B=0:I=FNB(C(C))
1670 REM SET VALUE TO I
1680 C(C)=0
1690 REM ADD VALUE
1700 C(C)=C(C)+I
1710 REM PRINT I$, VALUE
1720 P$=I$:GOSUB1900:M=1E8:GOSUB1860
1730 J=ABS(I):Z$=" ":Z=0
1740 D=INT(J/M):J=J-D*M:IFD=ZTHENP$=" ":GOTO1760
1750 Z$=",":Z=10:P$=CHR$(D+48)
1760 GOSUB1900:M=M/10:IFM=1E4THENP$=Z$:GOSUB1900
1770 IFM=10THENP$=".":GOSUB1900:Z=M
1780 IFM>=1GOTO1740
1790 IFI<0THENP$="CR":GOSUB1900: REM NOTE: NO SQUARE BRACKETS HERE
1800 GOTO1840
1810 REM PRE PRINT
1820 GOSUB1900
1830 REM NEW LINE
1840 P$=CHR$(13)+CHR$(10):GOTO1900
1850 REM COLUMN TAB
1860 IFP5THENPRINT#2,LEFT$(S$,41-C*10);
1870 IFL>70THENPRINTLEFT$(S$,41-C*10);
1880 P$=LEFT$(S$,25-LEN(I$))
1890 REM PRINT
1900 IFP5THENPRINT#2,P$;
1910 PRINTP$;:RETURN
```

## 202X Bar Graph Printer

John Easton of Toronto Ontario submitted this bar graph plotter for those who might need such a utility. The program here comes complete with demonstrations. It could be substantially shorter once all the cosmetics are removed.

```
100 REM VERSION 2.4     *     APRIL 11 '81
110 REM ************************************
120 REM *            BAR GRAPH             *
130 REM *    ADAPTED FROM  CBM NOTES BY    *
140 REM *           JOHN EASTON            *
150 REM *          FOR CHRISTIAN           *
160 REM * COMPUTER/BASED  COMMUNICATIONS *
170 REM *    44 DELMA DRIVE  -  TORONTO    *
180 REM ************************************
190 :
200 REM THIS FORM OF GRAPH IS KNOWN AS A GATT GRAPH
210 REM - (BAR GRAPHS ARE VERTICAL)
220 :
230 REM ************************************
240 REM *  CLEAR & INITIALIZE VARIABLES  *
250 REM ************************************
260 :
270 CLR : PRINT"[CLR]" : PRINT"[RVS] * BAR-GRAPH * " : PRINT
280 PRINT "[RVS ''''''''''''''''']BY JOHN EASTON * TORONTO "
290 PRINT : PRINT
300 PRINT "[RVS] NOTE > FOR USE WITH CBM 2022 PRINTER   ":PRINT:PRINT
310 PRINT "SELECT PRINTOUT FORMAT 'A' OR 'B'"
320 INPUT "OR ENTER '*' FOR EXAMPLES ";F$
330 IF F$="*" THEN GOSUB 1530 : F$=FF$
340 IF F$="A" OR F$="B" THEN 360
350 PRINT "[DN]" : GOTO 310
360 INPUT " OUTPUT TO PRINTER Y/N ";P$
370 REM FOLLOWING LINES SET OUTPUT FORMAT
380 LF=1 : DV=3 :                          REM OUTPUT FILE 1 - DEVICE 3 (SCREEN)
390 BAR=PEEK(213)-11:IF P$<>"Y"THEN 440: REM SET SCREEN WIDTH (40 OR 80 CHR)
400 BAR=68:DV=4:PRINT:                     REM PAGE WIDTH FOR OUTPUT ON DEVICE 4
410 IF PEEK(50003)=160 THEN LF=200:        REM SET OUTPUT CHANNEL HIGH FOR ROM 4
420 INPUT" WANT AN ENHANCED TITLE Y/N ";T$
430 T=129:IF T$="Y" THEN T=1 :             REM SET TITLE FOR ENHANCED MODE
440 A$(0)=""
450 A$(1)="[RVS]"+CHR$(165):               REM SET UP FRACTIONAL PRINTOUT ARRAY
460 A$(2)="[RVS]"+CHR$(180)
470 A$(3)="[RVS]"+CHR$(181)
480 A$(4)="[RVS]"+CHR$(161)
490 A$(5)=CHR$(182)
500 A$(6)=CHR$(170)
510 A$(7)=CHR$(167)
520 :
```

```
"[CLR]"  =  clear screen
"[HOME]" =  cursor home
"[UP]"   =  cursor up
"[DN]"   =  cursor down
"[CL]"   =  cursor left
"[CR]"   =  cursor right
"[RVS]"  =  reverse mode on
"[OFF]"  =  reverse mode off
"[ ' ]"  =  1 space
```

```
530 REM ************************************
540 REM *          SCREEN INPUT              *
550 REM ************************************
560 :
570 PRINT : INPUT "TITLE   *[CL CL CL]";T1$
580 IF T1$<>"*"  THEN 600
590 T1$=""
600 PRINT : INPUT "SUBTITLE   *[CL CL CL]";T2$
610 IF T2$<>"*"  THEN 630
620 T2$="" : GOTO 640
630 PRINT : INPUT "SUBTITLE AT 'T'OP OR 'B'OTTOM OF CHART   B[CL CL CL]";TT$
640 PRINT : INPUT "NUMBER OF ITEMS THIS GRAPH ";N
650 PRINT : INPUT "         LOWEST ITEM VALUE ";L
660 IF L<0 THEN PRINT "SORRY - NO NEGATIVE BARS"; : GOTO 650
670 L$=STR$(L) : LL=LEN(L$) :            REM CALCULATE LENGTH OF LOWEST VAL.
680 PRINT : INPUT "        HIGHEST ITEM VALUE ";H
690 H$=STR$(H) : HH=LEN(H$)
700 IF FLAG   THEN 870
710 IF F$="B" THEN BAR=BAR-HH : FLAG=1 : GOTO 870
720 GOTO 940
730 :
740 REM ************************************
750 REM *          OUT OF RANGE ?            *
760 REM ************************************
770 PRINT : PRINT " RANGE IS TOO LARGE TO REPRODUCE WITH   THIS FORMAT";
780 IF F$="B" THEN PRINT" - SELECT NEW 'R'ANGE" : GOTO 800
790 PRINT "-SELECT NEW 'R'ANGE OR 'F'ORMAT B"
800 GET B$ : IF B$="R"THEN 650
810 IF B$="F" THENPRINT"CHANGING TO FORMAT B FOR PRINTOUT ":F$="B":GOTO 650
820 GOTO 800
830 :
840 REM ************************************
850 REM *            FORMAT B                *
860 REM ************************************
870 MIN=(INT(H/(BAR)*100))/100 : MAX=(BAR/10)*L
880 MIN$=STR$(MIN) : MAX$=STR$(MAX)
890 PRINT : GOTO 1010
900 :
910 REM ************************************
920 REM *            FORMAT A                *
930 REM ************************************
940 PRINT
950 MIN=(INT(H/BAR*100))/100:MAX=(INT(L/LL*100))/100 : REM ROUND TO 2 DECIMALS
960 MIN$=STR$(MIN) : MAX$=STR$(MAX)
970 :
980 REM ************************************
990 REM *       SELECT SCALE VALUE           *
1000 REM ************************************
1010 IF MIN>MAX THEN 770
1020 PRINT " SELECT SCALE VALUE "MIN$" TO"MAX$" "
1030 PRINT " SMALLER SCALE NUMBER GIVES LARGER GRAPH "
1040 INPUT " SCALE  ";S
1050 IF S<MIN OR S>MAX THEN PRINT"ILLEGAL - PLEASE RE-ENTER SCALE" : GOTO1040
1060 :
```

```
1070 REM ***********************************
1080 REM *           DATA INPUT            *
1090 REM ***********************************
1100 PRINT "ENTER ITEMS IN FOLLOWING FORM :"
1110 PRINT : PRINT,"[RVS]ITEM DESC.[OFF] , [RVS]VALUE"
1120 DIM D$(N),V(N):                       REM DIMENSION INPUT ARRAY TO VAL.N
1130 FOR I=1TO N
1140 :  PRINT"ITEM"I, : INPUT"   *[CL CL CL]";D$(I),V(I)
1150 IF V(I)=>L AND V(I)<=H THEN 1180
1160 IF V(I)<L OR V(I)>H THEN PRINT"OUT OF RANGE - RE-ENTER VALUE -";
1170 INPUT"   #[CL CL CL]";V(I): GOTO 1150
1180 :  D$(I)=D$(I)+"          "
1190 :  D$(I)=LEFT$(D$(I),10)   :          REM PAD DESCRIPTOR TO 10 CHAR.
1200 NEXT
1210 :
1220 REM ***********************************
1230 REM *             OUTPUT              *
1240 REM ***********************************
1250 :
1260 SP$="" : FOR I=1TO BAR : SP$=SP$+" " : NEXT :  REM SET MAX BAR LENGTH
1270 PRINT "[CLR]"
1280 OPEN LF, DV
1290 PRINT#LF, TAB(11)CHR$(T);T1$
1300 IF TT$="B" THEN 1330
1310 PRINT#LF
1320 PRINT#LF, TAB(11);T2$
1330 PRINT#LF, : PRINT#LF
1340 FOR I=1 TO N
1350 REM FOLLOWING LINE ADJUSTS SCALE BY DIVIDING VALUE BY 'S' ADJUSTABLE VAL
1360 :  X=V(I)/S : Y=INT(X) : V$=STR$(V(I))
1370 IF F$="B" THEN 1400
1380 :  PRINT#LF, D$(I);" [RVS]"V$;
1390 :  PRINT#LF, "[RVS]"LEFT$(SP$,(Y-LEN(V$)));A$(8*(X-Y))"[DN]":GOTO 1430
1400 V$="          "+V$ : V$=RIGHT$(V$+" ",HH+1)
1410 :  PRINT#LF, D$(I);V$;
1420 :  PRINT#LF, "[RVS]"LEFT$(SP$,Y);A$(8*(X-Y))"[DN]"
1430 NEXT
1440 IF TT$<>"B" THEN 1470
1450 PRINT#LF : PRINT#LF,TAB(11);T2$
1460 PRINT#LF : PRINT#LF : PRINT#LF
1470 END
1480 :
1490 REM ***********************************
1500 REM *    OUTPUT FORMAT  EXAMPLES      *
1510 REM ***********************************
1520 :
1530 PRINT "[CLR DN DN ''''''''']OUTPUT FORMAT EXAMPLES"
1540 PRINT "--------------------------------------"
1550 PRINT
1560 PRINT "[RVS] FORMAT 'A' "
1570 PRINT
1580 PRINT : PRINT TAB(10) "TITLE (WITH ENHANCED OPTION)"
1590 PRINT : PRINT TAB(10) "SUB TITLE OPTIONAL"
1600 PRINT : PRINT : PRINT "DESCRIP'N [RVS] VALUE['''''''''']"
```

```
610 PRINT : PRINT          " 10 CHAR  [RVS] VALUE['''''''''''''']"
620 PRINT : PRINT TAB(10) "SUB TITLE OPTIONAL"
630 PRINT "-----------------------------------":PRINT:PRINT
640 PRINT "     *  SELECT FORMAT A OR B  *    "
650 PRINT "  OR PRESS RETURN FOR OTHER FORMAT  "
660 GET FF$ : IF FF$="" THEN 1660
670 IF FF$="A" OR FF$="B" THEN RETURN
680 PRINT "[CLR DN DN DN]"
690 PRINT
700 PRINT "[RVS] FORMAT 'B' "
710 PRINT
720 PRINT : PRINT TAB(10) "TITLE (WITH ENHANCED OPTION)"
730 PRINT : PRINT TAB(10) "SUB TITLE OPTIONAL"
740 PRINT : PRINT : PRINT "DESCRIP'N VALUE     [RVS '''']"
750 PRINT : PRINT          " 10 CHAR  VARIABLE  [RVS '''''''''''''''']"
760 PRINT : PRINT TAB(10) "SUB TITLE OPTIONAL"
770 GET FF$ : IF FF$="" THEN 1770
780 IF FF$="A" OR FF$="B" THEN RETURN
790 PRINT "[CLR DN DN DN]" : GOTO 1550
```

## Biocompatibility Program

This neat little number came from Joe Cannata of Medford, Long Island. It compares the biorythyms of two people at their time of birth. Commodore cannot be held responsible for any domestic differences this program may incur!

```
100 REM CALCULATE BIO COMPATIBILITY
110 DIM A(24),B(29),C(34),M(12)
120 DATA  100, 92.3, 81.6, 73.9, 65.2, 56.5, 47.8, 39.1, 30.4
130 DATA 21.7,   13,  4.3,  4.3,   13, 21.7, 30.4, 39.1
140 DATA 47.8, 56.5, 65.2, 73.9, 81.6, 92.3, 100
150 DATA  100,   93,   86,   79,   71,   64,   57, 50, 43, 36, 29, 21
160 DATA   14,    7,    0,    7,   14,   21,   29, 36, 43, 50, 57, 64, 71
170 DATA   79,   86,   93,  100
180 DATA  100,   94,   88,   82,   76,   70,   64, 58, 52, 46, 39, 33
190 DATA   27,   21,   15,    9,    3,    3,    9, 15, 21, 27, 33, 39, 46
200 DATA   52,   58,   64,   70,   76,   82,   88, 94, 100
210 DATA   31,   28,   31,   30,   31,   30,   31, 31,  30, 31, 30, 31
220 FOR Z=1 TO 24 : READA(Z) : NEXT
230 FOR Z=1 TO 29 : READB(Z) : NEXT
240 FOR Z=1 TO 34 : READC(Z) : NEXT
250 FOR Z=1 TO 12 : READM(Z) : NEXT
260 PRINT "[CLR]"
270 PRINT "BIORHYTHM COMPATIBILITY TEST"
280 PRINT "TYPE IN YOUR BIRTHDATE MM,DD,YY"
290 INPUT P, Q, R
300 GOSUB630
310 IF S=0 THEN 290
320 S1=S
330 PRINT : PRINT "TYPE OTHER PERSON'S BIRTHDATE"
340 INPUT P, Q, R
350 GOSUB630
360 IF S=0 THEN 330
370 S2=S
380 D9=ABS(S1-S2)
390 X2=A(D9-(INT(D9/23)*23)+1)
400 X3=B(D9-(INT(D9/28)*28)+1)
410 X4=C(D9-(INT(D9/33)*33)+1)
420 X5=(X2+X3+X4)/3
430 IF X5<25 THEN 470
440 IF X5<50 THEN 500
450 IF X5<75 THEN 530
460 GOTO550
470 PRINT : PRINT "WITH ONLY"X5"% COMPATIBILITY"
480 PRINT "FIND SOMEONE ELSE"
490 GOTO560
500 PRINT : PRINT "YOU SHOULD BE ABLE TO GET ALONG"
510 PRINT "WITH"X5"% COMPATIBILITY"
520 GOTO560
530 PRINT : PRINT"A RATING OF"X5"% IS NOT BAD"
```

```
540 GOTO560
550 PRINT : PRINT"MADE IN HEAVEN WITH"X5"%"
560 PRINT
570 PRINT "HERE ARE THE COMPATIBILITY BREAKDOWNS:"
580 PRINT "-----------------------------------"
590 PRINT "YOU ARE"X2"% PHYSICALLY"
600 PRINT "YOU ARE"X3"% EMOTIONALLY"
610 PRINT "YOU ARE"X4"% INTELLECTUALLY"
620 PRINT : PRINT : GOTO270
630 REM CALCULATE ELAPSED DAYS
640 IF R<1 THEN 860
650 IF R>99 THEN 860
660 IF INT(R)<>R THEN 860
670 S=R*365
680 S=S+((R-1)/4)
690 IF P<1 OR P>12 THEN 860
700 IF INT(P)<>P THEN 860
710 FOR I=1 TO P
720 S=S+M(I)
730 NEXTI
740 L=INT((R/4)*4)
750 IF P<3 THEN 770
760 S=S+1
770 IF Q<1 THEN 860
780 IF Q<>INT(Q) THEN 860
790 IF Q>M(P) THEN 810
800 GOTO840
810 IF L<>0 THEN 860
820 IF P<>2 THEN 860
830 IF Q>29 THEN 860
840 S=S+Q
850 RETURN
860 PRINT "[CLR]" : PRINT "INVALID DATE - TRY AGAIN"
870 S=0
880 RETURN
```

## Positioning For DATA READs

Everyone knows how RESTORE, READ and DATA statements operate. The first READ gets the first DATA element, and so on. RESTORE sets the READ pointer back to the beginning of text. But there is no command that allows positioning to a particular DATA line.

This could be useful if, for example, a DATA line were part of a subroutine. The only way to accomplish this in strict BASIC is to RESTORE and then issue enough READ commands to position to the desired data. This can be a pain!

RUN, CLR or RESTORE sets the DATA Read Pointer (address 62 and 63 decimal) back to $0400; the start of BASIC text. When a READ command is given, this pointer starts advancing through text looking for a 'DATA' line. If the pointer reaches the end of text before finding data, an ?OUT OF DATA ERROR occurs.

PET maintains another pointer that climbs up and down through text. This pointer is part of the CHRGET routine and essentially points at the code currently being executed. If this pointer (addresses 119 & 120) is transferred into the DATA Read Pointer, the next READ command would force a search to the next DATA line.

```
10 DATA FIRST, SECOND, THIRD
20 DATA FOURTH
30 READ A$, B$
40 POKE 62, PEEK(119) : POKE 63, PEEK(120)
50 READ A, B
60 PRINT A$, B$, A, B
70 DATA 1, 2, 3, 4
80 END
```

The READ command in line 30 gets "FIRST" into A$ and "SECOND" into B$, leaving the pointer pointing at "THIRD". Line 40 moves the pointer past line 10 and line 20 leaving it at some point in line 40. Since this is obviously not a DATA statement, the next READ causes an advance to line 70.

In summary, the POKEs of line 40 position to the next DATA statement in text.

## WHAT'S AVAILABLE ?!

Don White, Nepean Ontario

Recently, I have heard complaints that there is not much software available for the PET and CBM computers, that it is too difficult and expensive to add memory, that the number of peripherals is limited, etc., etc., etc. Therefore, over the next few issues of The Transactor I am going to try to provide a comprehensive list of software and hardware available. Most of the information will be culled from advertising in the computer magazines. If I miss anybody and you wish to be mentioned, please drop me a note with a brief description of your product. Users with comments, good or bad, about products they have purchased can drop me a line and I will attempt to include them when the product is mentioned.

Don White
Ottawa 6502 User Group
47 Ariel Court
NEPEAN, Ontario
K2H 8J1

This first offering consists of products that have been reviewed during the last 15 months in COMPUTE(COMP), KILOBAUD(KB) and CREATIVE COMPUTING(CC).

### Word Processor Software

Wordpro II (COMP 1, p 14) & Wordpro III (COMP 2, p 32), Commodore Business Machines, Santa Clara, CA 95051

Word Processor Program (COMP 1, p 17), Connecticut Microcomputers Inc., 150 Pocono Rd, Brookfield, CT 06804

Word Processor (COMP 1, p 19), Programma International, 3400 Wilshire Blvd, Los Angeles, CA 90010

Medit (COMP 2, p 30), Total Information Services, P.O.Box 921, Los Alamos, NM 87544

Textcast (COMP 2, p 30, p 100), Textcast, P.O.Box 2592, Chapel Hill, NC 27514

Papermate Command 60 (CC, Oct 80, p 168 & KB, Oct 80, p 13), AB Computers, 155 E Stump Road, Montgomeryville, PA 18936

### Printers

Axiom EX-801 & EX-820 (KB, Jan 80, p 14), Axiom Corporation, 5932 San Fernando Road, Glendale, CA 91202

CBM 2022 (CC, May 80, p 14 & CC, Dec 80, p 64), Commodore Business Machines, Santa Clara, CA 95051

Comprint 912 (CC, June 80, p 90 & KB, May 80, p 56), Computer Printers International, 340 E Middlefield Rd., Mountain View, CA 94043

M-100 Microprinter (COMP 4, p 66), Digiclocks, 3016 Oceanview Ave., Orange, CA 92665

XYMEC HY-Q 1000 (KB, July 80, P 6), XYMEC, 17791 Skypark Circle, Suite H, Irvine, CA 92714

## Languages

Waterloo BASIC (COMP 6, p 82), Computer Systems Group, University of Waterloo, Waterloo, Ontario N2L 3G1

Tiny PASCAL (COMP 9, p 124), Abacus Software, P.O.Box 7211, Grand Rapids, MI 49510

PET PILOT (CC, Feb 81, p 160 & KB, June 80, p 8), David Gromberg, Seven Gateview Court, San Francisco, CA 94116

PILOT (CC, Feb 81, p 158), Practical Applications, P.O.Box 4139, Foster City, CA 94404

PILOT (CC, Feb 81, p 158), Dr. Daley's Software, 425 Grove Ave., Berrien Springs, MI 49103

PILOT (CC, Feb 81, p 158), Peninsula School Software, Computer Project, Peninsula School, Peninsula Way, Menlo Park, CA 94025

## Music

KL-4M DAC (COMP 2, p 86) & Visible Music Monitor (COMP 8, p 110 & CC, Feb 81, p 18), AB Computers, 115 E Stump Rd., Montgomeryville, PA 18936

Petunia DAC & Petunia Player (COMP 1, p 42), HUH Electronics, 1429 Maple Street, San Mateo, CA 94402

K-1002-2 PETDAC (COMP 1, p 90) & K-1002-3C 4 Voice Music Software (CC, Oct 80, p 28), Micro Technology Unlimited, 2806 Hillsborough St., P.O.Box 12106, Raleigh, NC 27605

Music Box (CC, June 80, p 82), New England Electronics Co., 679 Highland Ave., Needham, MA 02194

RAM & ROM Adapters

PH-001 2114 RAM Adapter (COMP 5, p 81), Optimized Data Systems, P.O.Box 595, Placentia, CA 92670

Basic Switch (COMP 1, p 87 & KB, Mar 80, p 7), Applied Micro Systems, 3502 Home St., Misshawaka, Indiana 46544

Spacemaker II (KB, July 80, p 7), CGRS Microtech, P.O.Box 102, Langhorne, PA 19047

Dial-A-Rom (CC, Feb 81, p 154), Kobotek Systems Ltd., RR#1, Wolfville, N.S. B0P 1X0

Disk & Tape Systems

CGRS PEDISK (COMP 9, p 126), CGRS Microtech, P.O.Box 102, Laanghorne, PA 19047

D&R Cassette System (COMP 4, p 86 & KB, May 80, p 7), D&R Creative Systems, P.O.Box 402B, St. Clair Shores, MI 48080

Exatron Stringy Floppy (CC, Sept 80, p 60, p 190), Exatron Corporation, 181 Commercial St., Sunnyvale, CA 94086

Communications

TNW 488/103 Modem & PTERM Software (KB, Nov 80, p 12), TNW Corporation , 3351 Hancock St., San Diego, CA 92110

SOURCE Kit (KB, Nov 80, p 14), New England Electronics Co., 679 Highland Ave., Needham, MA 02194

Software - Computer Aided Instruction

Conduit, P.O Box 388, Iowa City, Iowa 52244 (CC, Jan 81, p 36)

Educational Activities Inc., P.O. Box 392, Freeport, NY 11520 (CC, Sept 80, p 68)

Milliken Publishing Co., St. Louis, Missouri (CC, Sept 80, p 56)

Microphys, 2048 Ford St., Brooklyn, NY 11229 (CC, Sept 80, p 192 & Oct 80, p 58)

Personal Software Inc., 1330 Bordeaux Drive, Sunnyvale, CA 94086

Program Design Inc., 11 Idar Court, Greenwich, CT 06830 (CC, Oct 80, p 58)

Tycom Associates, 68 Veelma Ave., Pittsfield, MA 01201 (CC, Jan 81, p 38)

## Software - Business

General Ledger, Accounts Payable, Accounts Receivable, Payroll (KB, Oct 80, p 13), CMS Software Systems, 5115 Menefee Dr., Dallas, TX 75227

JINSAM Data Base Management System (KB, Feb 81, p 12), Jini Micro Systems, Box 274, Bronx, NY 10463

Mailing List (COMP 4, p 79), Dr. Daley's Software, 425 Grove Ave., Berrien Springs, MI 49103

VISICALC (COMP 5, p 19), Personal Software Inc., 1330 Bordeaux Dr., Sunnyvale, CA 94086

## Software - Recreational

Temple of Apshai (COMP 1, p 86 & CC, Mar 80, p 40) & Morloc's Tower (COMP 2, p 5), Automated Simulations, P.O.Box 4232, Mountain View, CA 94040

Household Finance, Household Utilities (KB, May 80, p 8), Creative Software, P.O.Box 4030, Mountain View, CA 94040

Software for 3G Lightpen (KB, Oct 80,p 13), Quill Software, 2512 Roblar Lane, Santa Clara, CA 95051

Microsail (CC, Dec 80, p 24) & Batter Up! (COMP 2, p 98), Hayden Book Company Inc., 50 Essex Street, Rochelle Parks, NJ 07662

Talking Calculator (COMP 5, p 39), Programma International, 2908 Naomi St., Burbank, CA 91504

Jagdstaffel (CC, Dec 80, p 22), Discovery Games, 936 W Highway 36, St. Paul, MN 55113

Softpac-1 (CC, Sept 80, p 194), Competitive Software, 21650 Maple Glen Dr., Edwardsburg, MI 49112

Bridge Challenger (COMP 1, p 91), Personal Software, 592 Weddell Dr., Sunnyvale, CA 94086

## Software - Utilities, Assemblers, etc

PROGANAL (CC, Jan 81, p 158 & KB, Oct 80, p 12), Benson Greene, 210 Fifth Ave., New York, NY 10010

EDIT, SNAPSHOT (KB, Mar 81, p 10), California Software Associates, Box 969, Laguna Beach, CA 92652

MAE (COMP 3, p 93 & KB, Aug 80, p 20) & PET RABBIT (COMP 3, p 94), Eastern House Software, 3239 Linda Dr., Winston-Salem, NC 27106

MONJANA/1 (KB, Jan 81, p 10), Elcomp Publishing, 3873L Schaefer Ave., Chino, CA 91710

PRO-KIT#1 (KB, Mar 81, p 8), Intex Datalog Ltd., Eaglescliffe Industrial Estate, Eaglescliffe, Stockton-on-Tees, Cleveland TS16 OPN, England

EP-2A-79 EPROM Programmer Software (COMP 2, p 89), Optimal Technology Inc., VA 22936

BIG KBD (CC, Oct 80, p 170), PROGRAMMERS TOOLKIT (COMP 2, p 4 & CC, July 80, p 24 & KB, Apr 80, P 34), PROGRAMMERS DISK-O-PRO (COMP 8, p 112 & CC, Feb 81, p 156), Skyles Electric Works, 231 E South Whisman Rd., Mountain View, CA 94041

Miscellaneous

3G Lightpen (CC, Mar 80, p 161 & KB, June 80, p 9), 3G Company, Rt.3, Box 28A, Gaston, OR 97119

Plexi-Vue High Contrast Screen (COMP 2, p 99 & CC, Sept 80, p 192), Competitive Software, 21650 Maple Glen Dr., Edwardsburg, MI 49112

Anti-Glare Screen (CC, Sept 80, p 192), Pf Research, 866 Hummingbird Dr., San Jose, California

Prestodigitizer (COMP 3, p 56 & CC, Nov 80, p 168 & KB, Mar 80, p 8), Innovision, P.O.Box 1317, Los Altos, CA 94022

New-Cursor (COMP 1, p 79, p 82), International Technical Systems, P.O.Box 264, Woodbridge, VA 22194

PETSET 1a (COMP 1, p 31), AIM-16 & SADI (KB, Nov 80, p 12), Connecticut Microccomputers Inc., 150 Pocono Rd., Brookfield, CT 06804

THS224 Real Time 1/3 Octave Audio Analyser (KB, Jan 81, p 48), Eventide Clockworks, 265 West 54th St., New York, NY 10019

Mathematics & Foreign Language ROMs (CC, Feb 81, p 154 & KB, Dec 80, p 7), Kobetek Systems Ltd., RR#1, Wolfville, N.S. B0P 1X0

Full Sized Keyboard (CC, Jan 81, p 160), Century Research & Marketing, 4815 West 77th St., Minneapolis, Minn 55435

PIE Parallel Interfacing Element (KB, Nov 80, p 15), LEM Data Products, P.O.Box 1080, Columbia, MD 21044

PET BIBLIOGRAPHY — Don White, Nepean Ontario

The following is a bibliography of PET related articles published in Creative Computing and Kilobaud Microcomputing from January 1980 to February 1981.

## Kilobaud Microcomputing 1980

### January

Page 116 The Metamorphosis of a 'Custom' PET
126 Darkroom Master
14 PET-pourri

### February

112 Development of a Text-Handling Program
158 Add a Digital Tape-Index Counter to the PET

### March

59 The Comprint Printer
132 The Keyed-up PET
7 PET-pourri

### April

34 The BASIC Programmers Toolkit
172 The PET Librarian
186 Indexing for the PET
9 PET-pourri

### May

7 PET-pourri

### June

58 PET I/O Expander
122 Computer Survival Course for Kids
8 PET-pourri

### July

22 Get Your PET on the IEEE 488 Bus
36 New Additions to the Commodore Line
84 PET Pen
7 PET-pourri

### August

96 PET I/O Expander
134 Get Your PET on the IEEE 488 Bus
168 A 'Personable' Calendar
20 PET-pourri

September

| | |
|---|---|
| 26 | New Product, New Philosophies |
| 30 | Write Self-Modifying PET Programs |
| 32 | Memory Expansion Candidates |
| 34 | PET Machine-Language Masquerade |
| 36 | Add a Reset Button to Any PET |
| 40 | The Phantom Tape Drive |
| 44 | Get Your PET on the IEEE 488 Bus |
| 56 | PET I/O Expander |
| 60 | The PET/CMC/H14 Connection |
| 10 | PET-pourri |

October

| | |
|---|---|
| 88 | PET Mini Monitor |
| 180 | Betting on Old POKEy |
| 12 | PET-pourri |

November

| | |
|---|---|
| 173 | Microcomputer Hardware for the Handicapped |
| 12 | PET-pourri |

December

| | |
|---|---|
| 218 | Give Character To Your PET Printer |
| 7 | PET-pourri |

## Kilobaud Microcomputing 1981

January

| | |
|---|---|
| 48 | Real Time Spectrum Analyser |
| 78 | Scramble |
| 188 | Second Cassette Interface |
| 10 | PET-pourri |

February

| | |
|---|---|
| 56 | London Computer Club a Huge Success |
| 72 | Portrait of a Dynamic French Company |
| 12 | PET--pourri |

## Creative Computing 1980

January

148 Personal Electronic Transactions

February

112 Blackbox for the PET
156 Personal Electronic Transactions

March

60 PET as a Remote Terminal
160 Personal Electronic Transactions

April

22 Atari in Perspective

May

14 PET 2022 Line Printer
152 Personal Electronic Transactions

June

52 Neelco's Music Box for the PET
90 Comprint 912 Printer
154 Making a PET

July

24 The BASIC Programmers Toolkit
28 BASICs Comparison Chart

Septeember

98 Computer Countdown
190 Personal Electronic Transactions

October

26 Sound Advice
168 Personal Electronic Transactions

November

164 Personal Electronic Transactions

December

22 Softwar, Hardware and Otherware for Christmas
26 Christmas Buyers Guide
60 The PET
64 The CBM 2022 Smart Printer
70 Comparative Evaluation of Basic Systems

## Creative Computing 1981

January

24 No PET Peeves
156 Personal Electronic Transactions

February

18 Music Editors for Small Computers
154 Personal Electronic Transactions

The following is a bibliography of PET related articles in COMPUTE! from FALL 1979 to February 1981.

## COMPUTE! 1979

### Fall 1979, Issue 1


13 Three Word Processors
24 Microcomputers for Nuclear Instrumentation
29 Tokens Aren't Just For Subways
31 PETSET 1a
40 Flying With PET PILOT
42 Petunia & Petunia Player
42 Teacher, Computers and the Classroom
65 The Evolution of a Magazine
68 PET in Transition
71 A Commodore Perspective
76 Retrofitting ROMs
78 Screen Print Routine
78 PET Resources
79 Review: New-Cursor
80 Cassette Format Revisited
82 Review: New-Cursor
84 Trace for the PET
86 32K Programs Arrive
87 Review: The BASIC Switch
89 Non-Stop PET, Old and New
89 Un-crashing on Upgrade ROM Computers
90 Review: 8-Bit Digital to Analog Converter
91 Review: Bridge Challenger
93 Using Direct Access Files with the 2040


## COMPUTE! 1980

### January/February, Issue 2


3 A Visit to Commodore
4 The Programmers Toolkit
5 Micro Quest/Simulation
5 Space Invaders
6 Comprehensive and Massive PET Manual
7 Interview with Dr. Chip
18 Memory Partition of BASIC Workspace
29 Word Processors
41 BASIC Memory Map
43 The Learning Lab
44 Micros and the Handicapped
78 Computer Programs and Your Ethics
81 Lower Case Descension on the Commodore 2022 Printer
82 Saving Memory in Large Programs
82 Apparent Malfunction of the < Key
82 Yes Nova Scotia, There is a 4 ROM PET
82 The Deadly Linefeed
86 4-Part Music System for PET

87 Using Direct Access Files with the 2040
90 Null Return Simulation for PET Users
93 A few entry points, original/upgrade ROM
93 Plotting With the CBM 2022 Printer
94 Inside the 2040 Disk Drive
96 PET Programs on Tape Exchange
98 Review: Batter Up!
99 Review: Plexi-Vue
100 Review: Text Cast

March/April, Issue 3

8 Dr. Chip
10 A Preview of Commodore's New Disk BASIC 4.0
15 Enhancing Commodore's WordPro II
18 File Conversion on the Commodore 2040 Drive
23 Using the GET Statement on the PET
34 UTINSEL: Enabling Utilities
41 Manual Alphabet Tutorial on a PET
51 The Learning Lab
56 Review: The Prestodigitizer
60 Light Pen Selection from Large Menus
63 The Consumer Computer
81 Fix for Lower Case Descension Program
81 Comments on Null Return Simulation
82 Cheep Print
88 Direct Screen Input
88 No CB2 Sound?
90 A Versatile Serial Printer Interface
92 Ramblin': Bar Charts
93 Review: MAE, A PET Disk-based Macro Assembler
94 Review: The PET Rabbit
95 PET Programs on Tape Exchange
96 Memo to Machine Language Programmers

May/June, Issue 4

14 The CBM 8032
37 Review: PET and the IEEE 488 Bus
42 Big Files on a Small Computer
48 Using the 2nd Cassette Buffer to Increase Memory
53 Enhancing Commodores WordPro III
58 Algebraic Input for the PET
59 PET Data Copier
63 Cross-Reference for the PET
69 The Learning Lab
74 The PET 'ANSWER BOX' Program for Teachers
77 PET GET With Flashing Cursor
86 D&R Cassette System
89 PETting With a Joystick
90 PET and the Dual Joysticks
104 Block Access Method Map for a 2040 Disk Drive
109 Ramblin': Hard Copy Without a Printer
111 Cheep Print, Part 2
115 Relocate PET Monitor Almost Anywhere
119 PET Programs on Tape Exchange

July/August, Issue 5


23 Programming in BASIC with the Power of FORTRAN
28 Computers and the Handicapped
69 Butterfield Reports: The 8032
73 Plotting with the 2022 Printer
81 Un-New
81 Review: PH-001 2114 RAM Adapter
83 Disk ID Changer(Note: Do NOT use this)
85 Shift Work
86 ML Code for Appending Disk Files
89 Mixing BASIC & Machine Language
92 After the Monitor's Moved
94 Fitting Machine Language into the PET
96 Using Disk Overlays on the PET
99 Joystick Revised
103 Ramblin': The User Port


September/October, Issue 6


18 Teaching Basic Academic Skills
40 Computers for the Handicapped
42 Hat In The Ring
82 Waterloo Structured BASIC for the PET
86 TelePET
89 WordPro Converter
90 Quadra-PET
92 Update to Disk ID Changer, Issue 5
94 Variable-Field-Length Random Access Files
98 Flexible GET for the PET
100 ROM-antic Thoughts
102 Converting ASCII Files to PET BASIC
104 Compactor
110 A few entry point, original/upgrade/4.0 ROM
112 Feed Your PET Some APPLESOFT


November/December, Issue 6


18 Music and the Personal Computer
22 Micros and the Handicapped
24 Small Computers and Small Libraries
30 Efficiency with Subroutines
78 Basic CBM 8010 Modem Routines
80 Programmer's Notes for the CBM 8032
84 Keyprint
88 PET 4.0 ROM Routines
92 BASIC 4.0 Memory Map
94 Algebraic Expression Input, Version 2
96 Defining a Function Whilst Running a Program
98 Machine Language Addressing Modes
104 Visible Memory Printer Dump
110 Disk Lister
115 Commodore Dealers Form Cooperative

## COMPUTE! 1981

### January, Issue 8


10 An Interview with Dr. Chip
12 VIC!
22 Spend Time, Save Money!
26 Micros with the Handicapped
28 Kids for Computers
32 The Mysterious and Unpredictable RND
38 CURSOR Classifications Revisited
44 Odds & Ends re PET Cassette Tape
92 The Screen Squeeze Fix for CBM 8000
96 Hooray for SYS
102 Scanning the Stack
108 Review: The PET Revealed
108 Review: Library of PET Subroutines
110 Review: Visible Music Monitor
112 Review: Disk-O-Pro
114 Correcting Alignment on your PET Tape
118 Spooling for PET with 2040 Disk Drive
118 Variable Dump for New ROM PETs
120 The 32K Bug
121 The Ideal ML Save for the PET
122 PET Metronome
123 The IEEE Bus - Standing Room Only
124 PET/CBM IEEE Bus Error


### February, Issue 9


16 LED A Line-Oriented Text Editor
34 The Mysterious and Unpredictable RND
54 Basic Math for Fun and Profit
60 PET Spelling Lessons your Students Can Prepare
97 Contour Plotting
103 Relocate
104 Mixing & Matching Commodore Disk Systems
109 Memory Calendar
114 Crash Prevention for the PET
116 Machine Language Printer Command
118 Odds & Ends on PET/CBM Files
120 Three PET Tricks
124 Review: PASCAL on the PET
126 Review: The PEDISK from CGRS Microtech
127 Review: Disk Operating System for the PEDISK

## Print Using

Wnat to format numbers before printing ? Just use this neat little subroutine by Jim Butterfield. Pass it a value 'V' and it gives you back 'V$', all nicely padded with leading spaces and trailing zeroes. V1 and V2 are the the number of digits to the left and right of the decimal place. An overflow will return all asterisks. (Also in Compute)

```
100 REM DEMO PROGRAM FOR SUBROUTINE
110 FORJ=1TO20
120 REM V IS VALUE TO BE FORMATTED
130 V=EXP(RND(1)*14-6)*SGN(RND(1)-.2)
140 REM V1 IS # OF DIGITS LEFT OF .
150 REM V2 IS # OF DECIMAL PLACES (RIGHT OF .)
160 V1=4:V2=0:GOSUB50000:PRINTV$;" ";
170 V1=3:V2=1:GOSUB50000:PRINTV$;" ";
180 V1=3:V2=4:GOSUB50000:PRINTV$
190 NEXTJ
200 END
50000 REM 'USING' ARRANGE IN COLUMNS
50010 REM V IS VALUE; V1.V2 PRINTS
50020 V4=INT(V*10^V2+.5)
50030 V$=RIGHT$("          "+STR$(V4),V1+V2+1):Q$=V$
50040 IF V2<1 GOTO50080
50050 FORV5=V1+2TOV1+V2+1:IF ASC(MID$(V$,V5))<48THENNEXTV5
50060 V6=V5-V1-1
50070 V$=MID$(V$,V6,V1+1)+LEFT$(".00000",V6)+MID$(V$,V5)
50080 IF ASC(V$)>47 THEN V$=LEFT$("**********",V1+V2+2+(V2=0))
50090 RETURN
```